Project 1: Dams

ENGR 151

Released: September 12, 2024

Due: September 26, 2024, 11:59 P.M.

1 Introduction

A dam is a barrier constructed to hold water and form a reservoir so that the water can be used for various purposes, such as consumption, irrigation, and power generation. You have just been hired by a large power utility company which is looking to diversify its energy portfolio by adding hydroelectric power from a new dam. In this project, you will be involved in the planning of this construction!

2 Creating a Visualization

Your company has already selected a suitable valley in which they hope to build the dam. They have prepared some useful data about this site in the provided `project1_data.mat` file. In particular, the file contains three fields:

- `x`: a one-dimensional array which defines all the x-coordinates at which height measurements were taken.
- `y`: a one-dimensional array which defines all the y-coordinates at which height measurements were taken.
- `valley`: a two-dimensional array for which each entry is the topographic height of the valley at that coordinate.

The company wants you to produce a visualization of the valley and reservoir similar to the one shown in Figure 1. Thankfully, since you are new to the job, your colleague has pointed you to some internal documentation on how to accomplish this.

2.1 Landscape Visualization Documentation

1. Read about the MATLAB `meshgrid` command in Appendix Section A. If you like, you can also read MATLAB's official documentation on the command.
2. Create a MATLAB function called `reservoir_visualization` which takes in one argument, `water_height`, that represents the water level to be shown in the visualization. The function should return one variable `water_surface`, which will contain the array of values representing the surface of the water. Then within the function...
3. Create meshgrid arrays `X` and `Y` from the provided vectors `x` and `y`.
4. Create another array which represents the height of the water at each coordinate. This should be the same size as the `X` and `Y` arrays, and initially, it is okay for it to simply populate water everywhere at the desired `water_height`.

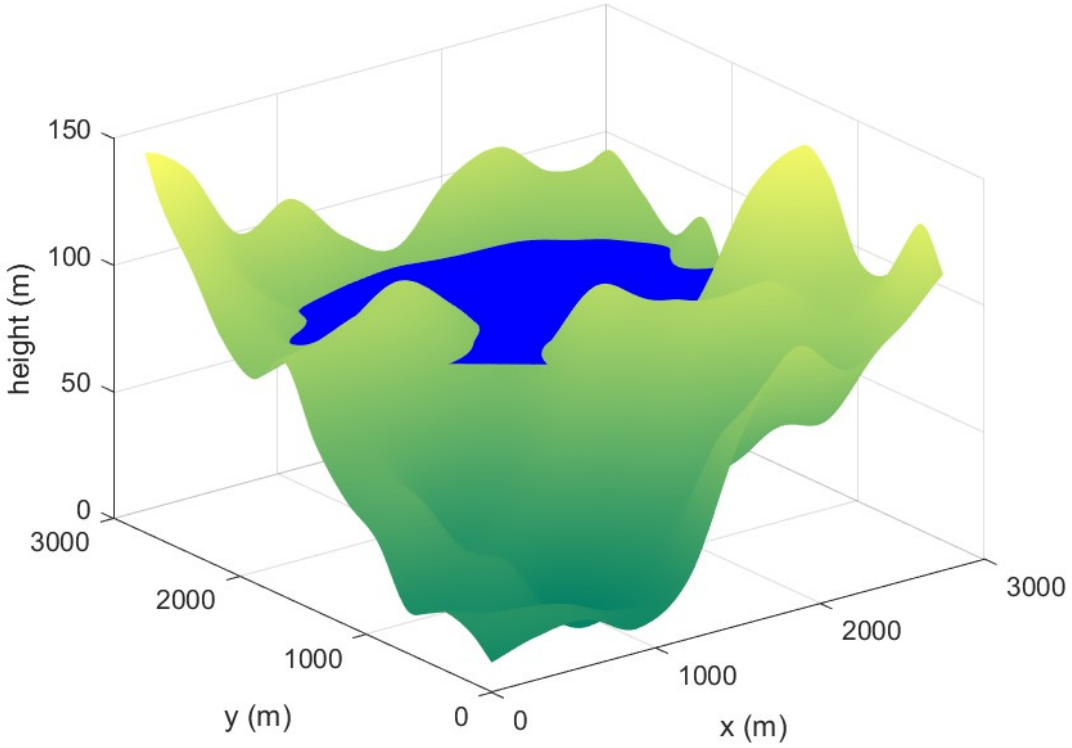

Figure 1: Visualization of the valley and reservoir with the water level at 80 meters. You may use this image as reference for what a correct output should look like (at this water level).

5. At this point, you can plot your `water` and `valley` arrays like this:

```
% Open a new MATLAB Figure
figure;
% Define the set of colors to be used in this plot
colormap summer;
% Plot the valley
surf(X,Y,valley,EdgeColor="none");
% Configure the current plot so that new plot commands do not clear it
hold on;
% Plot the water and make it blue
surf(X,Y,water,EdgeColor="none",FaceColor="Blue");
hold off;
```

The `EdgeColor="none"` argument passed into both plot commands just makes MATLAB plot without drawing black lines where the edges of the meshgrid are, which in such a high resolution meshgrid as these are, just make the whole surface look black. Your visualization will not look right yet, it probably looks something like Figure 2.

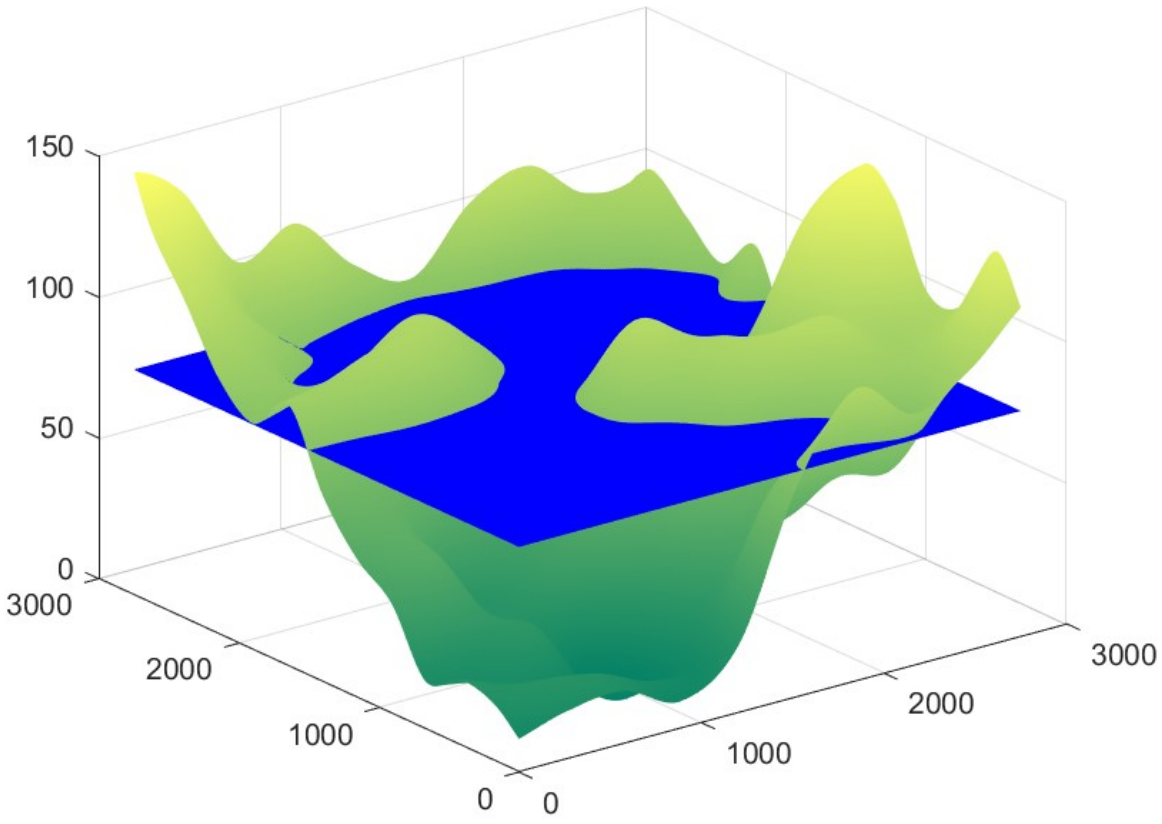

Figure 2: Unfinished visualization of the reservoir with water populated everywhere.

6. Now you need a way to tell MATLAB where it should not draw water. A handy trick you can use for this is that if a value in an array being plotted is set to `NaN` (stands for **N**ot **a** **N**umber) then

MATLAB will simply plot nothing where it would've plotted that value. You could go through every value in your water array and manually set each value you do not want plotted to `NaN`, but that would be tedious and take a long time. Instead, try to take the two conditions below and turn them into logical statements on the meshgrids you have in your program:

- There should be no water beneath any part of the valley
- There should be no water below the proposed location of the dam, which is to be constructed on the line given by:

$$y = -0.8x + 2200$$

7. Finally, add appropriate axis labels and units, and save your figure to a file. Don't forget to return the array of the water's surface!

3 Calculating the Volume of the Reservoir

The next thing you've been assigned to do is to calculate a good approximation for the volume of water that will be in the reservoir for any water level. Specifically, you must write a MATLAB function called `reservoir_volume`, which again takes in one parameter, `water_height`. Then, using the data on the valley provided in `p1_data.mat`, along with the same logic you used in the first part of the project to isolate the area where water will be, form an estimate for the volume of the water in the reservoir. **Your estimate should be given in units of millions of cubic meters.**

You helpful colleague mentions that a senior engineer in the department already performed these estimates and even recorded the results below in Table 1. Unfortunately, they left the company and did not document their work, so now it must be redone.

Water Height (m)	Reservoir Volume (millions of cubic meters)
70	3.61
75	11.24
80	21.17

Table 1: Estimates for the volume of water in the reservoir at various water levels. Your estimates should match these to within 10%.

HINT: Remember, this valley and the water are both defined by meshgrids, which have constant distance between points in both the x and y directions, denoted here as Δx and Δy. For any location in the reservoir, the water has a certain depth D, and so the approximate volume of water at that location alone is:

$$V_{(x,y)} = \Delta x \cdot \Delta y \cdot D$$

The an approximation for the volume of water in the reservoir can be attained by summing the volume of water at each location.

4 Calculating the Maximum Power Output of the Dam

The term "hydroelectric power" refers to electricity generated by harnessing the power of moving water. Dams can be used to produce a great deal of hydroelectric power. Water falls from the reservoir behind the dam, through a turbine, which then spins and produces electricity. The amount of power generated in this fashion can be estimated with knowledge of just a few quantities:

$$P = m \cdot g \cdot H \cdot \eta$$

Where

- m is the mass flow rate of water through the system. That is, the number of kilograms of water passing through the turbine each second.
- g is the acceleration due to gravity. For this project, use $g = 9.81\ \mathrm{kg\,s^{-2}}$.
- H is the vertical distance that the water falls in meters.
- η is the total efficiency of all components of the system. This is a unitless quantity.

Your final task is to calculate an upper bound on the amount of power the dam can produce for a given water level, the efficiency of the system, and the flow rate of water. Specifically, write a MATLAB function called `max_power` which takes in three arguments:

- The water height in the reservoir (m)
- The system efficiency
- The mass flow rate of water through the dam ($\mathrm{kg\,s^{-1}}$)

This function should return the power the dam outputs **in gigawatts**, assuming all water falls from the water level in the reservoir to the lowest point in the valley (regardless of whether that point is actually below the dam or not). As it happens, the same engineer who calculated estimates for the reservoir volume also calculated some estimates for the maximum power output of the dam. They still failed to document their work, but you can use their results to compare your own work against.

Water Height, h	System Efficiency, η	Mass Flow Rate, m	Max Power Output, P
75	0.7	1.2×10^6	0.62
75	0.7	1.5×10^6	0.77
80	0.7	1.5×10^6	0.82

Table 2: Estimates for the the maximum power output of the reservoir (in gigawatts) for several values of the input variables. Your estimates should match these to within 10%.

5 Grading

The project should be submitted to Project 1 on Canvas. You should submit the following files:

- `reservoir_visualization.m`
- `reservoir_volume.m`
- `max_power.m`

The Matlab autograder will evaluate the results of your submission and provide a maximum score of 100 points, broken down by task as shown below.

- Task 1: 40 points
- Task 2: 30 points
- Task 3: 30 points

Please take a look at the file "Style_Expectations.pdf" in Project 1 Google Drive folder for information on the style and commenting of your code. Try to start following the best practices outlined within. For the first two projects, you will not be graded on the style of your code, but starting with Project 3, it will be part of your score. If in doubt, seek help in office hours / Piazza!

Appendices

A MATLAB Meshgrids

You have likely known about the Cartesian coordinate plane for some time, and have seen many examples of how useful it is in your mathematics courses over the years. Thus, it is not really surprising that scientists and engineers often have a need of representing and calculating with the Cartesian coordinate plane (and many other coordinate systems) in their programs. But such a representation is not trivial to create. How can one represent a plane with infinite width and height — where even the space between two infinitesimally close points has an infinite number of points in it — in finite computer memory?

The short answer is: you can't! However, we generally only need a finite portion of the coordinate system we are using, and even within that portion, we only need a finite resolution of points. With these considerations in mind, representations for coordinate grids can be built on a computer. One very popular representation is called a "meshgrid", and MATLAB has built-in functionality for meshgrids. To learn more about how meshgrids work and how to use them, consider the following example.

First, let's create two vectors, named x and y, which will define the intervals of the X- and Y-axis we want to represent in our meshgrid.

```
>>  x = 0:4; % x = [0, 1, 2, 3, 4]
>>  y = 0:5; % y = [0, 1, 2, 3, 4, 5]
```

Now let's define our meshgrid using MATLAB's `meshgrid` command.

```
>>  [X,Y] = meshgrid(x,y);
```

Let's examine these outputs `X` and `Y` which we got back.

```
>>  X =

     0     1     2     3     4
     0     1     2     3     4
     0     1     2     3     4
     0     1     2     3     4
     0     1     2     3     4
     0     1     2     3     4

>>  Y =

     0     0     0     0     0
     1     1     1     1     1
     2     2     2     2     2
     3     3     3     3     3
     4     4     4     4     4
     5     5     5     5     5
```

Notice each row of `X` is a copy of `x` and each column of `Y` is a copy of `y`. Notice also that both `X` and `Y` have the same number of rows as there are entries in `y`, and the same number of columns as there are entries in `x`. To make what is happening here easier to see, `Y` has been reprinted below, flipped vertically.

```
>>  Y =

    5     5     5     5     5
    4     4     4     4     4
    3     3     3     3     3
    2     2     2     2     2
```

```
    1     1     1     1     1
    0     0     0     0     0
```

Now, looking at `X` and the new version of `Y`, we can see that `X` contains the x-coordinate of every point in the interval defined by `x` and `y`, and likewise `Y` contains the y-coordinate of every point in that interval.

As cool as this is, it gets better! `X` and `Y` are just matrices, and like any other matrices in MATLAB, they can be operated upon. Consider:

```
>> my_first_inequality = Y<X

my_first_inequality =

  6×5 logical array

   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   1
   0   0   0   1   1
   0   0   1   1   1
   0   1   1   1   1
```

Now `my_first_inequality` is a logical matrix (a matrix of boolean elements), with a 1 everywhere $y < x$ on our interval of the Cartesian plane! Of course, similar operations can be performed with other operators like $=$, $\leq$, $>$, $\geq$, etc. Give it a try!

Note: In practice, you should not manually flip the `Y` array of your meshgrid as we have done here for illustrative purposes. The proper orientation of the `Y` array is handled internally within MATLAB.

Now that you've learned how to create meshgrid arrays in MATLAB, and a little bit about how they can be used, you're ready to get started on the first part of the project. Good luck!