Chapter 22

Flatland Challenge

Contents

22.1 The Challenge

You will write a program to drive your Neato on the floor of the classroom in a way that physically realizes the method of steepest ascent. The mountain you will "climb" is defined as follows:

$$z = f(x, y) = 8\exp\left(-1.5(x + y - .75)^2 - .5(x - y - .75)^2\right) + 6\exp\left(-(x + .75)^2 - (y + .25)^2\right),$$

where x, y, and z are measured in meters. At minimum, you should be able to start your Neato at $(0.2, -0.3)$, pointing in the $+y$ direction, and work your way to the top.

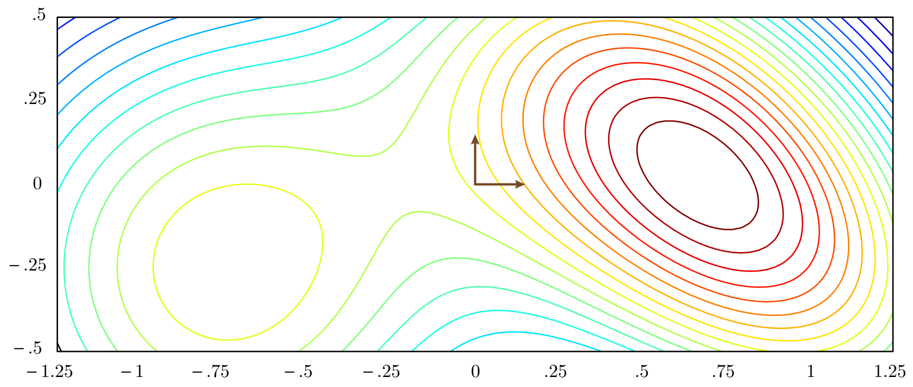

Figure 22.1: Contour map of Flatland function.

Exercise 22.1

1. Decompose this problem. What are the steps involved? You should have a decomposition that

clearly explains the process of driving the Neato along a discrete approximation to the path of steepest ascent.

2. Now develop the code for the Neato. Start by writing pseudocode and/or comments with the steps you developed. Then build that into your code. It may help to review class 11 and your Bridge of Doom code.

3. Now drive your Neato! We recommend you start with the simulator to check to see that your code works properly. You can use the command `>> neatov2.connect(); neatov2.setPositionAndOrientation(x, y, theta);` to place your Neato in the correct starting location and orientation. The variables `x` and `y` are the x and y starting positions in meters, and `theta` is the heading in radians, where 0 is with the robot facing the +x direction. To start the simulator and plot the function contours, you can use the flatland starter code, `flatlandStarter.m`, which is in the same MATLAB drive folder as `neatov2`.

   Remember, full instructions for using the simulator and the robot hardware are available on the Meeto Your Neato page.

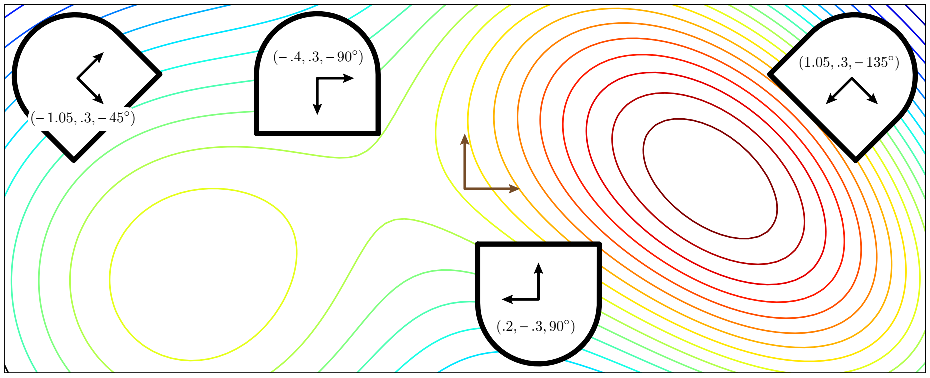

Figure 22.2: Contour map of Flatland function with different starting position/orientations for the Neatos. In (x, y, θ) form, these starting position/orientations are: $(.2, -.3, 90°)$, $(1.05, .3, -135°)$, $(-1.05, .3, -45°)$, $(-.4, .3, -90°)$. Note that x and y are in meters, and that $\theta = 0°$ corresponds to the Neato travelling in the $+x$ direction.

22.2 Deliverables

You should turn in a short write up to canvas with the following deliverables.

1. An explanation of your method including any relevant equations. You may use your *pseudocode*, but please do not use your actual code here.
2. A contour plot of the function with your theoretical path of gradient ascent.
3. A screen capture video of the simulated Neato moving along the path of gradient ascent, starting from $(.2, -.3)$ with the Neato facing in the $+y$ direction.
4. A video of your real life Neato traversing the path of gradient ascent, starting from $(.2, -.3)$ with the Neato facing in the $+y$ direction.
5. A link to your MATLAB code.
6. Try your code starting at $(-1.05, .3)$, with the Neato facing $-45°$ (a southeast heading). Does it still work? Why or why not? How might you edit your code to work for all points?

7. Edit your code to work for any starting point.
8. Write a brief description of how the behavior of gradient ascent changes based on the current point you are at.

22.3 Optional Flatland Extension A: Continuous Approach

If we want to follow the continuous path of steepest ascent, we should set the velocity of the Neato proportional to the gradient, $\mathbf{r}'(t) = \alpha\nabla f$, where α is a parameter that we can choose depending on how fast we want the Neato to drive.

Exercise 22.2

1. How does the speed of the Neato depend on α and ∇f? If α is a constant, how fast is the Neato moving when it reaches the "top" of the hill? How would we choose α if we wanted the Neato to move with constant forward velocity v?
2. In terms of coordinates x and y, this approach is equivalent to defining the following set of ordinary differential equations, of the type that you encountered in ModSim:

$$\begin{aligned}\frac{dx}{dt} &= \alpha\frac{\partial f}{\partial x}\\ \frac{dy}{dt} &= \alpha\frac{\partial f}{\partial y},\end{aligned}$$

   where $x(0)$ and $y(0)$ represent the initial position of the Neato. You can either solve these in MATHEMATICA using DSolve, in MATLAB using dsolve, or by hand if you are very familiar with solving differential equations.
3. Implement this and drive your Neato in a continuous path uphill!

22.4 Optional Flatland Extension B: Accelerating Convergence

Gradient ascent often leads to paths with many zig-zags. One way to reduce this effect is to not simply take the gradient as the ascent direction at each step, but rather take a combination of the current gradient and prior movement direction; i.e., include a momentum term into the movement direction. Specifically, the *accelerated gradient ascent* algorithm still takes the form

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \lambda_i \mathbf{z}_i,$$

where $\mathbf{z}_i$ is the direction of ascent. However, now the direction is chosen as

$$\mathbf{z}_i = \nabla f(\mathbf{r}_i) + \gamma_i \mathbf{z}_{i-1}, \tag{22.1}$$

where γ_i is a parameter that controls the weight of the combination of the current gradient direction and the previous movement direction. The videos here and here cover various methods to accelerate convergence. Try implementing one of these momentum method to see if helps your Neato reach the peak of Flatland more quickly.

Finally, if you are feeling super ambitious, you can try to implement a *conjugate gradient ascent* method. For these methods, the new direction is chosen as in (22.1), except that the γ_i is not a constant parameter! Rather, it is a quantity that can be chosen at each iteration using information such as the lengths of the gradients or the Hessian matrix evaluated at recently visited points. See the first few pages of this survey paper if you want to try out a method from this class.