QEA2

Flatlands

Daniel Sudzilouski, Sparsh Gupta

Olin College of Engineering

March 8, 2023

Methodology

Gradient Ascent

The "flatlands mountain" equation on which the Neato drives is defined as follows:

$$z = f(x, y) = 16e^{\frac{-x^2}{2} - \frac{y^2}{2} - \frac{xy}{2}} + 4e^{-(x+1.5)^2 - (y+2.5)^2}$$

To determine the maximum of the above function of many variables, we implement the Gradient Ascent Algorithm which takes steps in the direction of the gradient. So, the height of the flatland is described by $z = f(r)$, where r is the position vector in this plane.

The next points are determined by the gradient ascent algorithm when we begin at the initial point r_0. Here, the starting coordinates of the Neato are at $r_0 = (-2, 0)$. The next points are given by the equation:

$$r_{i+1} = r_i + \lambda_i \nabla f(r_i), i = 0, 1, 2, ...,$$

where λ_i is the relative step size we are taking in the direction of the gradient.

Evaluating the step size λ_i at every iteration to maximize the function f (*exact line search*) is too computationally expensive and therefore, we implement *Backtracking line search* which rather picks the step size at each iteration that increases the function value by at least $\alpha\lambda_i||\nabla f(r_i)||^2$ where α is an arbitrary threshold parameter with constraints $0 < \alpha < 0.5$. In our implementation, α is set equal to a value of 0.1

Therefore, the initial value of step size λ_i is set equal to 0.25 and then it is decayed over iterations using a decay factor β ($0 < \beta < 1.0$), which is set equal to 0.5 here, by the equation $\lambda_i = \beta\lambda_i$.

A pseudocode implementation of the above backtracking line search algorithm looks like this:

```
Set: $\lambda_i = 0.25$
While: $f(r_i + \lambda_i \nabla f(r_i)) < f(r_i) + \alpha\lambda_i||\nabla f(r_i)||^2$
    $\lambda_i = \beta\lambda_i$
Return: $\lambda_i$
```

Neato Simulation

The Neato faces +y-axis at its initial position in the simulation. This corresponds to an initial angle of $\pi/2$ for the Neato. For determining the next angle (θ) to move towards, the following is used:

$$\theta = atan(\frac{y_{i+1} - y_i}{x_{i+1} - x_i})$$

where (x, y) are the coordinates of the Neato.

To determine the amount of distance to travel in a straight line once after calculating the angle to rotate, the following equation is used:

$$distance = \sqrt{(x_i - y_i)^2 + (x_{i+1} - y_{i+1})^2}$$

where (x, y) are the coordinates of the Neato.

Contour Plot

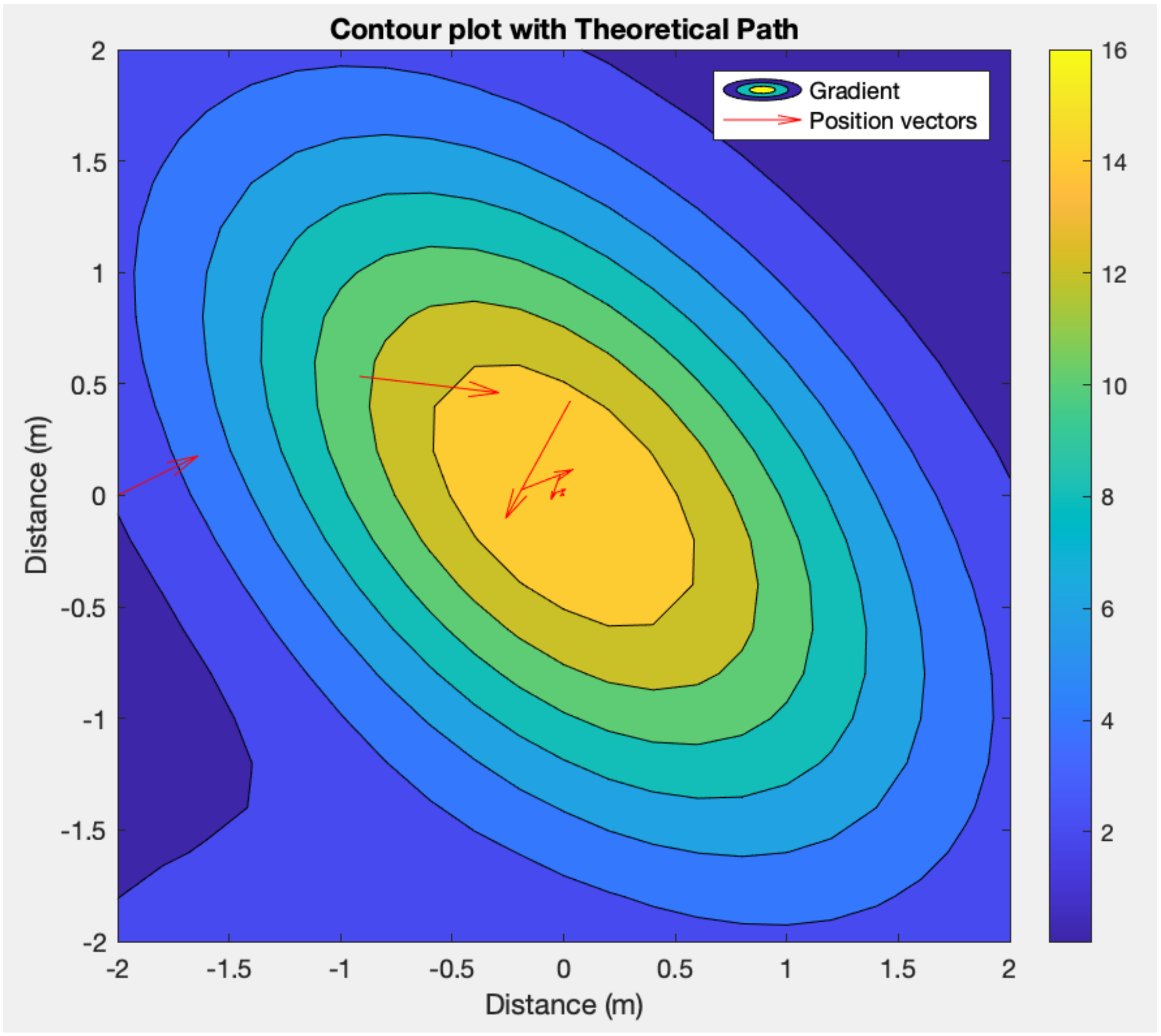

Figure 1: A contour plot of the function with the theoretical path of gradient ascent starting from (-2,0).

Simulated Neato Screenshot

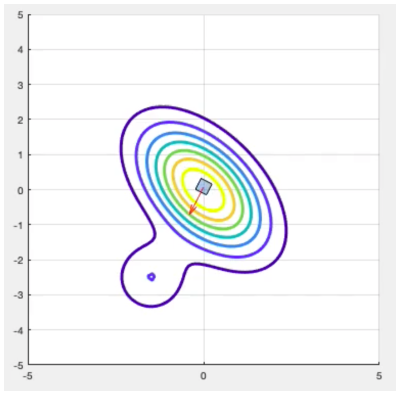

Figure 2: A screenshot of the simulated Neato on the gradient flatland starting from (-2,0).

Optimized Implementation: Simulation with different starting points

We tried various examples of setting alternate starting points in the Neato simulator. For example, the coordinates (1, 0) and (-3, 1) also result in the Neato ending towards (0,0) in the simulation which is the point in the gradient we are supposed to reach.

This is a result of using an optimized approach of determining when to stop when the Neato reaches a critical point. We set this tolerance = 0.1 and the gradient ascent algorithm is executed until the gradient value at the points stays higher than this tolerance level. The Neato stops when the gradient is less than the tolerance value and this results in finding the most optimal path.

However, this results in a longer runtime to find the most accurate finishing point on the gradient. So, this optimization algorithm is a trade-off between accuracy vs efficiency. We chose the tolerance value based on multiple tests and determined that this is the most optimal value to have a greater accuracy than normal without compromising much on the efficiency.

Screen Capture Video, Neato Video and Code

Click here to watch the screen capture video of the simulated Neato.

Click here to watch the robot in action as it traverses the Flatlands.

Click here to view the repository of code to conquer Flatlands.