Flatland Project

Dongim Lee, Zara Coakley

Equation:
z = f (x, y) = 8*exp(-1.5*(x + y - .75)^2 - .5*(x - y - .75)^2) + 6*exp(-(x + .75)^2 - (y + .25)^2),

Gradient:
F_x = -6*e^(-(x+3/4)^2-(y+1/4)^2)*(2*x+3/2)-8*e^(-(y-x+3/4)^2/2-3*(x+y-3/4)^2/2)*(4*x+2*y-3)
F_y = -6*e^(-(x+3/4)^2-(y+1/4)^2)*(2*y+1/2)-8*e^(-(y-x+3/4)^2/2-3*(x+y-3/4)^2/2)*(2*x+4*y-3/2)

Methodology:
The algorithm uses the gradient of the function to determine the direction of steepest ascent, and then adjusts the step size accordingly to converge towards the maximum. By iteratively updating the position and orientation while considering the gradient information, the algorithm effectively climbs the mountain, eventually reaching a local maximum.

Iteration:

- Calculate the gradient of the function at the current position (r_i).
- Update the position using the gradient ascent formula: r_i = r_i + step_size * grad_i.
- Update the step size (lambda) using the decay rate: step_size = delta * step_size.
- Calculate the orientation change (theta_i) required to move from the current position to the next one.
    - $\tan(\theta) = \Delta y/\Delta x \rightarrow \theta = \arctan(\Delta y/\Delta x)$
- Calculate the time needed to rotate and then move straight to the next position.
    - To rotate
        - $\omega=(vR-vL)/d$
        - vL = -vR = wheel_speed
        - $\omega=(2*wheel\_speed)/d$
        - $\Delta\theta=\omega t \rightarrow t = \Delta\theta/\omega = \Delta\theta$ / (2*wheel_speed/d) = $\Delta\theta$ * d / (2*wheel_speed)
    - To move straight
        - $v=(vL+vR)/2$
        - vL = vR = wheel_speed
        - v = wheel_speed
        - $\Delta x = \sqrt{(\Delta x^2+\Delta y^2)}$
        - $\Delta x=vt \rightarrow t = \Delta x/v = \sqrt{(\Delta x^2+\Delta y^2)}$ / wheel_speed
- Append the updated position (r_i), orientation (theta_i), and time (time_rotate, time_straight) to their respective history lists.

Assuming that left and right wheel velocity are the same (when moving forward), the time history provides a path that the Neato robot can follow to climb the simulated mountain. It does this by iterating through the time history, alternating between adjusting the heading and moving forward.

Link to video of the simulated and real-life Neato:
https://www.youtube.com/watch?v=w4vaio0ntZ8

Contour Map:

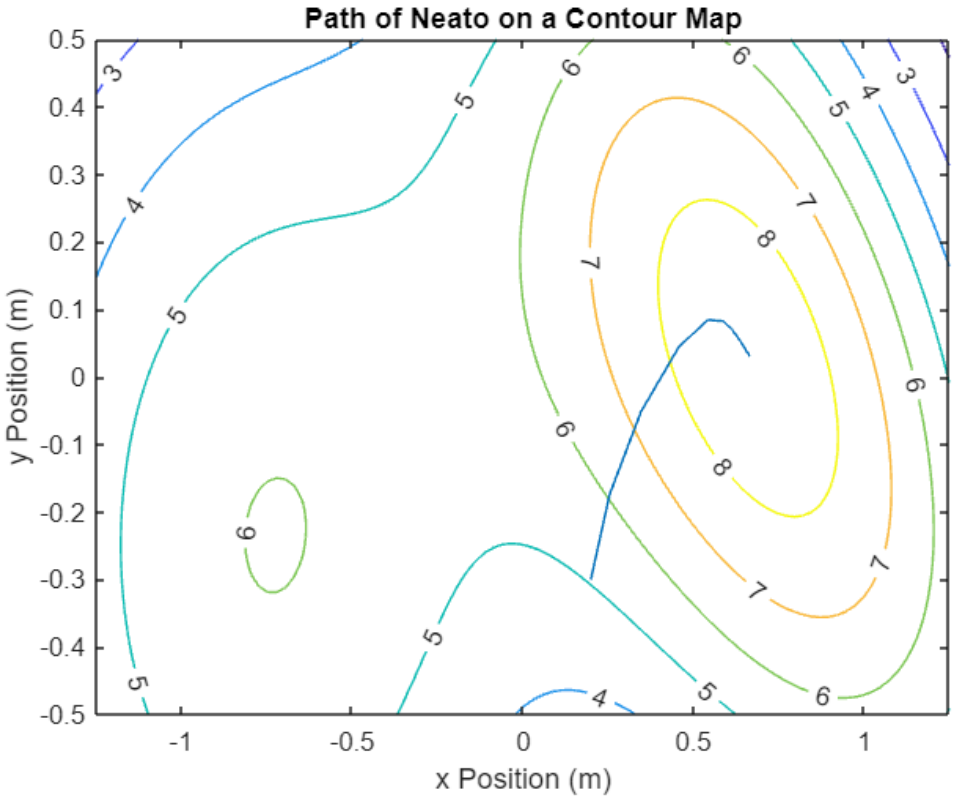

Figure 1. A contour map showing the path of Neato following our gradient ascent algorithm to arrive at the maximum point of the function z = f (x, y) = 8*exp(-1.5*(x + y - .75)^2 - .5*(x - y - .75)^2) + 6*exp(-(x + .75)^2 - (y + .25)^2).

Link to matlab code:
https://drive.google.com/file/d/1gykNAqYBywB9GarVULtmsCj7Pzi7L7-7/view?usp=drive_link

How the gradient ascent behavior changes based on the point you're at:
6. When we set the starting point at (-1.05, 0.3), it didn't work at first. As we looked into it, we realized that our code had a mistake because some of our rotation time values were negative. We hadn't noticed earlier because from the original point the Neato only turns clockwise, and the negative time values only appeared when the Neato needed to turn counterclockwise. To address this, we refined our code to handle cases where the time duration became less than zero. In such cases, we programmed the Neato to rotate counterclockwise (the default rotation direction being clockwise) for the absolute value of the time duration.

This arose due to our method of driving the Neato, which involved setting specific durations for rotation and straight movement. Since we calculated our time based on the theta value ($t=\Delta\theta/\omega$), when theta changed sign, so did time. Consequently, rotation durations became negative when the Neato attempted to rotate left.

8. The gradient points in the direction of greatest increase at the location where the Neato is. This means that it will head in the direction that moves it upward most quickly in its immediate vicinity. This doesn't mean that it will find the absolute maximum point because once it finds its way to a local max all routes of direction from that point would take it downward, so it will never leave that maximum.