Mount Doom: Write-Up

Matt Brucker and Prava Dhulipalla

I. EXERCISES

A. *Exercise 1*

The initial gradient at r_0 was (0, -9). The gradient at r_1 was $(\frac{63}{8}.\frac{9}{16})$. The gradient at r_2 is $(\frac{873}{320}, \frac{67}{320})$. The distance between the two points is $|r_1 - r_0| = \lambda_0|\nabla f(r_0)|$.

B. *Exercise 2*

To rotate the NEATO so that it points in the right direction, the current direction and the gradient have to be known (both vectors of x,y). The angle between them is represented by the equation:

$$cos^{-1}(\frac{currentdirection \cdot gradient}{\|currentdirection\| * \|gradient\|})$$

To implement this, time was calculated using this equation:

$$t_{rot} = \frac{\theta}{\omega}$$

Where theta is found in Equation 1 and omega is an arbitrary value. This calculated theta and chosen omega can be used to determine the time it would take to turn from the current direction towards the direction of the gradient. Using the pause command, the NEATO can move the correct angle if it turns the correct amount of seconds.

C. *Exercise 3*

To find the time it would take to travel a certain distance, the following equation is used:

$$t = \frac{d}{s}$$

Where d is the distance (calculated using the gradient) and s is the speed (given by the problem, 0.1 m/s - or, in the units of this problem, .328 ft/s).

D. *Exercise 4*

We thought about it.

E. Exercise 5

α represents a multiplier on the speed of the robot when moving. If α is a constant, then the speed of the robot would be zero when it reaches the top of the hill. In order to make the robot drive at a constant speed, then alpha must be inversely proportional to $|\nabla f|$ - which we tried to do, but we were not successful at :(.

F. Exercise 6

In order to find the position coordinates (x,y) with respect to time, the following differential equations must be solved:
$\frac{\mathrm{d}x}{\mathrm{d}t} = \alpha \, \frac{\partial f}{\partial x} \; \frac{\mathrm{d}y}{\mathrm{d}t} = \alpha \, \frac{\partial f}{\partial y}$
This step is important because the original gradient equation is not in respect to time, but instead is in respect to x and y. Although this worked for the discrete portion of this assignment, for the parametrized/continuous part, the equations needed to be in respect to time. Thus, the above differential equations needed to be solved. This was done using MATLAB's dsolve function to find the resultant equations.

G. Exercise 7

We ended up making everything work by making α increase proportionally with time, and by giving it a magic number multiplier that make it go a reasonable speed.

H. Exercise 8

For a 30° roll in ϕ:

$$\begin{bmatrix} 0 \\ 1\frac{}{2} \\ \sqrt{3}\frac{}{2} \end{bmatrix}$$

For a 45° pitch in θ:

$$\begin{bmatrix} -1\frac{}{\sqrt{2}} \\ 0 \\ -1\frac{}{\sqrt{2}} \end{bmatrix}$$

For a 15° yaw in ψ:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

I. Exercise 9

Yaw is always the same, despite movement. This is probably because the gravity vector will always point perpendicular relative to the Earth/ground, so the matrix will always have 1 in the z direction.

J. *Exercise 10*

Only two angles are needed to completely specify the gravity vector in space - pitch and roll.

K. *Exercise 11*

To completely define the NEATO in three dimensional space, three angles are needed - pitch, roll, and yaw. Thus, there is not enough information from the measurement of the gravity vector to completely define the orientation of the NEATO.

L. *Exercise 12*

Although we never kept track of the values we got, we DEFINITELY did the question. We swear on our NEATO's lives.

M. *Exercise 13*

For a 45° pitch then roll:

$$\begin{bmatrix} -\sqrt{2}_{\overline{2}} \\ 1_{\overline{2}} \\ 1_{\overline{2}} \end{bmatrix}$$

N. *Exercise 14*

roll: 79.84°

pitch: -29.74°

O. *Exercise 15*

$$\begin{bmatrix} 3.5934 \\ 3.3830 \\ 4.0372 \end{bmatrix}$$

P. *Exercise 16*

If the robot is currently pointing in the direction of the gradient, then $G_y = 0$. At that point, the slope is equal to $\frac{G_z^2}{1-G_z^2}$.

Q. *Exercise 17*

The rotation is about the z-axis (of the NEATO). In order to determine the angle it has to rotate, the following equation must be used:

$$\theta = tan^{-1}(-\frac{G_y}{G_x})$$

R. Exercise 18

Pseudocode: 1. Read accelerometer values 2. Calculate the θ by which to rotate 3. Rotate the robot 4. Calculate the slope using G_z 5. Multiply the slope by λ. 6. Move by that amount. 7. If we're at a point that we think is the top, stop. Otherwise, repeat.

S. Exercise 19

Check out our code on the Githubs: It's pretty NEATO

T. Exercise 20

I (Matt) already did this with Connor for Bridge of Death, so we just reused our code for that. You can find it here.

U. Exercise 21

Check out our code on the Githubs: link

V. Exercise 22

The position appears to be decently accurate, although it took us some work to get to that point. There seemed to be a lot of error in the z-direction, which was probably caused by an inaccurate value of the angle at which the robot lies "flat." Other inaccuracies are probably caused by the fact that, in our position calculations, we assumed the robot was rotating on a plane, when in reality it just wasn't. Also, there were definitely points where the robot was running into the wall, and that's going to make the plot think it's going forward when it really isn't.

II. Process Documentation

Overall, our process of climbing up Mount Doom was mostly trial and error (emphasis on the error). The process of implementing the discrete algorithm for flatland was relatively straightforward, and we were able to accomplish that fairly quickly. Once we got to the continuous algorithm, things got a bit more complicated. The main issue we encountered was how to solve the differential equations with an α that would give a constant robot speed. However, although we were fairly sure on what to do, we couldn't figure out any way to make it work in MATLAB, and because of time constraints, we decided to just make α proportional to time and add in a "fudge factor" to make it work - which actually worked pretty darn well. This is where things started to go downhill (or, I guess you could say, down-Mount Doom). When it came to the actual climbing up Mount Doom, we didn't really understand what to do because there was a lot of complicated stuff that was difficult to wrap our heads around. Eventually, though, Paul explained everything to us and we realized that it was much simpler than we thought. We originally tried to make it climb up using actual gradient ascent with the slope and a λ, but that didn't work because the step size varied too much. So, instead, we just gave it a constant step size, which was pretty okay. However, we weren't able to come up with a way too make it stop at the top, although we did try a bunch of janky methods, none of which worked. The mapping was actually fairly straightforward; we decided to just record the data once and plot everything based on that, because it was much simpler than trying to plot data while running. It turns out that we're bad at that, so it took a bit longer than expected, but we got done! Then, we already had a pretty solid idea of how to plot things based on our work on Thursday, so there were only a few issues with plotting, all of which we were able to resolve fairly quickly. Overall, we understood some things pretty well, but there were definitely a lot of spots where we just had to put in magic numbers and use trial and error to make things work sufficiently.

III. Robot Videos

Check out our videos: Vid 1: The robot running on flatland. Vid 2: The robot climbing up Mount Doom.