Space Engineering 3
Assignment 2
1st May 2016

Global Navigation Satellite Systems

Lydia Drabsch
311217591
ldra3557@uni.sydney.edu.au

STUDENT PLAGIARISM: COURSE WORK - POLICY AND PROCEDURE COMPLIANCE STATEMENT

INDIVIDUAL / COLLABORATIVE WORK

I/We certify that:

(1) I/We have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*;

(2) I/We understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to the University commencing proceedings against me/us for potential student misconduct under Chapter 8 of the *University of Sydney By-Law 1999* (as amended);

(3) this Work is substantially my/our own, and to the extent that any part of this Work is not my/our own I/we have indicated that it is not my/our own by Acknowledging the Source of that part or those parts of the Work.

Name(s): Lydia Drabsch

Signature(s):
Lydia Drabsch

Date: 24/3/16

Contents

List of Figures

List of Tables

1. Introduction

Each mainQN.m file has a section called 'User Input' where the animations and state plots can be turned on/off. Also the timestep and the number of days to simulate are defined. The default settings are dt = 100 seconds and days = 1.

2. Grid Based Robot Motion Planning

2.1 Introduction

output of each function as well as a discussion of your understanding of each section and how you implemented the relevant algorithms.

2.2 Generate Grid

The function **generateGrid** calculates the size of the matrix that reflects the discretised map by *mapDim*/*cellDim* and creates the matrix as *map*. The start and end node are calculated using the provided function *pos2cell* and set in *map*. The map is displayed by the predefined function **showmap**, see Figure 4.1.

Inputs:

- *mapDim*: The width and height of the map in meters.
- *cellDim*: The required width and height of each cell on the grid.
- *startPos*: The x and y position of the start point in meters from the centre of the grid.
- *goalPos*: The x and y position of the goal point in meters from the centre of the grid.

Outputs:

- *map*: The discretised global map with start and goal node defined

2.3 Generate Obstacles

The function **generateObstacles** defines the cells on the map where the rover cannot travel due to predefined obstacles. The map reflects the configuration space of the rover, therefore the radius of the rovers footprint was added to the radius of each obstacle. The x domain of each obstacle was defined in meters discretised by . Using the conic equation for a circle Eq(1), the x domain was to calculate the upper and lower edge of the obstacle in meters.

$$y_i = y_c \pm \sqrt{r^2 - (x_i - x_c)^2} \tag{1}$$

Where x_i is the discretised domain, (x_c, y_c) is the centre of the obstacle, r is the new radius of the obstacle and y_i is the discretised upper and lower edge of the obstacle. The coordinates (x_i, y_i) were converted to indices in *map* by **pos2cell**. For each x_i index, the coordinates between y_i^+ and y_i^- were set as an obstacle in *map*. The output *map* is displayed using **showmap**, see Figure 4.2.

Inputs:

- *map*: The discretised global map from **generategrid**.
- *obstacles*: The x,y coordinates and the radius of each obstacle
- *cellDim*: The required width and height of each cell on the grid.

- *mapDim*: The width and height of the map in meters.
- global *ROVER_FOOTPRINT_RADIUS*: The radius of the footprint of the rover in meters.

Outputs:

- *map*: The discretised global map with start node, goal node and obstacles defined.

2.4 Footprint Points

The function **getRoverFootprintPoints** uses the built-in function **rangesearch** to identify points in *pointCloud* that fall within the rover's footprint radius from the centre of a cell.

2.5 Least Squares Fit of Plane

The function **fitplane**
Inputs:

- *points*:

Outputs:

- n: vector normal to the plane
- p: average vector

2.6 Step Hazard Evaluation

The function **stepHazardEval**

2.7 Pitch Hazard Evaluation

The function **pitchHazardEval**

2.8 Roughness Hazard Evaluation

The function **roughnessHazardEval**

2.9 Identify Neighbours

The function **getNeighbours**

2.10 Dijkstra Algorithm

The function **dijkstraBody**

2.11 A* Algorithm

The function **aStarBpdy**

3. Question 2

3.1 Introduction

The final GEO orbit is defined as having a period of one sidereal day, 23 hours 56 minutes 4.0916 seconds. Using Kepler's third law Eq(2) and the vis-viva law Eq(3) the semi-major axis and velocity of the circular orbit were calculated, see Table 3.1.

$$a = \sqrt[1/3]{\frac{\mu \mathbb{P}^2}{4\pi^2}} \tag{2}$$

$$v = \sqrt{\frac{\mu}{a}} \tag{3}$$

The satellite parameters in the park orbit are

Table 3.1: text

Orbital Parameter	Initial Value	Final Value
Semi-major axis	6655937 m	
Period		86164.0916 s
Velocity		
Eccentricity	0	0
Inclination angle	-28.5°	0°
RAAN	0°	
Argument of Perigee	0°	
Mean Anomaly	0°	free
Epoch	0 s	free

3.2 Cost and Constraints

What are constraints - why are they important.
The constraints on this optimisation problem defines the final GEO orbit. They are The radius and velocity are enforced by requiring the ratio between the

$$J = |\Delta v_1| + |\Delta v_2| \tag{4}$$

3.3 Local frame of satellite

The local frame of the satellite where the burn is applied

3.4 Methodology

The Hessian of the Lagrangian $H_{\mathcal{L}} = \nabla^2_{xx}\mathcal{L}$
BFGS method. Approximation of the Hessian update

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k - \frac{\boldsymbol{H}_k s_k s_k^T \boldsymbol{H}_k}{s_k^T \boldsymbol{H}_k s_k} + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T s_k} \tag{5}$$

$$s_k = x_{k+1} - x_k \tag{6}$$

$$y_k = \nabla f_{k+1} - \nabla f_k \tag{7}$$

4. Appendix A: Grid Based Robot Motion Planning

Figure 4.1: Output from Section 2.2 Generate Grid. The start node is in blue and the goal node is in red

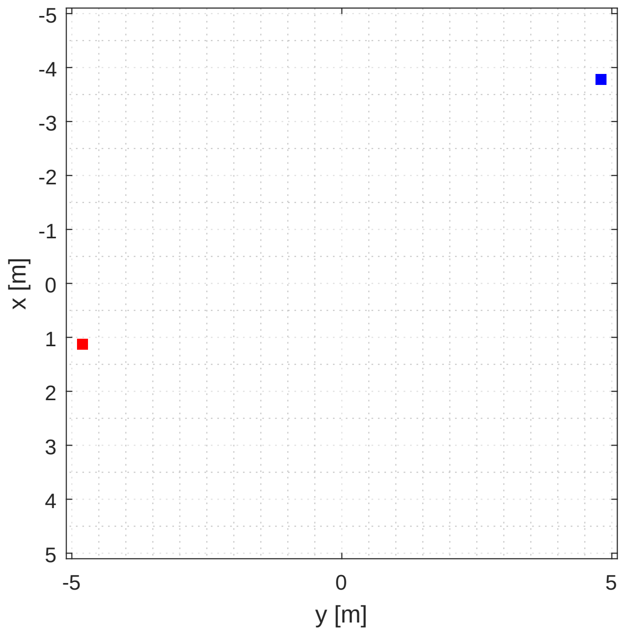
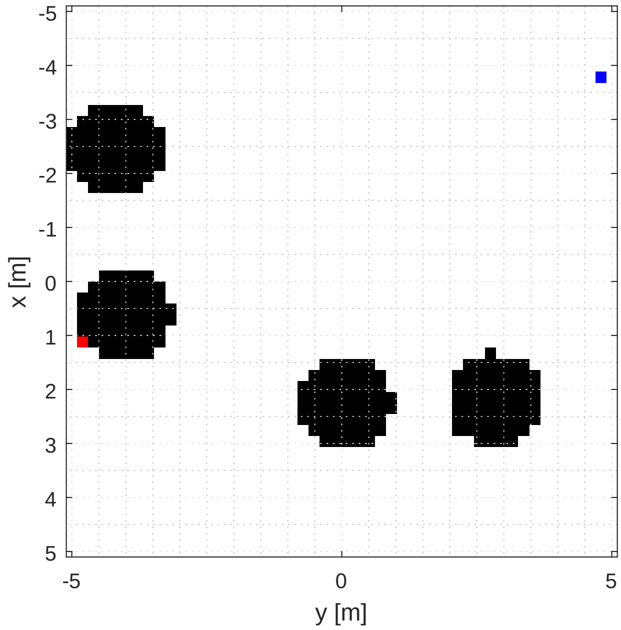


Figure 4.2: Output from Section 2.3 Generate Obstacles. The start node is in blue, the goal node is in red and obstacles are black



5. Appendix B: Question 2