Satellite Formation Flight Navigation Using the Clohessy-Wiltshire Equations

A Senior Project

presented to

the Faculty of the Aerospace Engineering Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Anna Kimmich

June, 2009

Table of Contents

Table of Figures

Abstract

This report presents the design and simulation of a satellite navigation program using Matlab. The program allows any number of independent satellites to navigate in a co-planar cluster formation in a circular orbit with no input besides its location coordinates from a simulated GPS unit and the relative location of other satellites from a simulated on-board sensor. This navigation program uses a potential field-like function to calculate the desired satellite position within the formation. It uses the Clohessy-Wiltshire equations to calculate the impulsive maneuvers needed to achieve and maintain the formation.

Nomenclature

r	=	distance, *km*
v	=	velocity, *km/sec*
t_f	=	final time, *sec*

Subscripts

0	=	target location
c	=	chaser
t	=	target

I. Introduction

SATELLITE formation flight is a developing field of technology. Traditional single-unit satellite systems are limited by launch vehicle capacity, and constellations with widely spaced units cannot accomplish all the same missions as formations, due to the distance between satellites[3]. Satellites in formation can be used to overcome these limitations. The difference between satellites in a formation and a constellation is the type of control system used. Satellites in formation are all within a relatively small area and need to continuously update their orbits to keep in formation and avoid collisions, especially when other units in the formation change their positions. This necessitates orbit corrections to be made autonomously by the satellite units rather than the groundstation operators. Satellites in a constellation are spread out, for example the GPS constellation is spaced out around the entire globe. The relative positions of constellations do not need to be as exact as formations, which is why their orbits are only adjusted on a time scale of weeks or months.

Formations of satellites can be used for Earth observation and for astronomy, including interferometry. NASA's New Millennium Program Earth Orbiter-1 (EO-1) spacecraft successfully demonstrated formation flight by autonomously following Landsat 7 on an Earth observation mission. A cluster of small satellites could possibly replace single large satellites in the future and could be modular, which would lower risk to the mission if one failed, and it would be cheaper to repair and upgrade a single unit than to rebuild and relaunch an entire large satellite. Another potential use for satellites with formation GNC systems is for eliminating space junk. The system could allow another satellite to maneuver very close to any other orbiting object for observation and tracking.

The objective of this senior project is to develop a guidance and navigation program that uses a satellite's own location, determined by GPS, and the range and location of its adjacent satellites, determined by onboard sensors, to calculate and move it to its desired orbit autonomously. The use of local rather than global knowledge means each satellite with this guidance system can operate independently and can achieve a formation with any other satellite or orbiting object. This program is designed to keep the satellites in a planar cluster formation. The original objective

of this project was to build the program entirely with Simulink, however a Matlab script was used instead, due to technical difficulties. The results are recorded in an animated plot showing the location of each satellite in ECI coordinates.

II. Design

This simulation models three satellites, two of which use a formation flight GNC program. The simulation was made as a Matlab script which calls several Matlab functions and a dynamics model in Simulink. It is designed to be used for autonomous flight after the satellites have been placed on coplanar circular orbits near enough to each other that they can sense each other, but far enough that collisions are not imminent. After they are placed in this initial set of orbits, the formation GNC system assumes control and brings the satellites close to their final formation and keeps them there. The location of individual satellites within the final formation will change with time, but the overall formation will be maintained.

A. Formation Description

This formation navigation system is designed to keep three satellites in a coplanar cluster. One satellite is the "lead" and uses conventional rather than formation navigation. It has been included to avoid formation drift. In this case the term "lead" means its navigation is independent of the other satellites, not that it is located at the front of the formation. Both of the other satellites in this simulation use the formation GNC software. Each one of them considers itself to be the "chaser" and adjusts its relative position accordingly. It senses the other satellites, but does not communicate with any of them. The benefit of this is that it does not need to rely on other units in the formation meaning it enables a satellite using this software to fly in formation with any object it can sense. This use of local knowledge means that the formation is egalitarian in the sense that the satellites place themselves according to which other satellites are closest without regard for any particular unit.

B. Simulation

The simulation uses a simulated GPS-determined location as the initial location for each satellite. The simulation uses the vehicle dynamics function in Appendix A to continuously update the lead vehicle location throughout the run time. The lead vehicle has a circular LEO orbit. Each satellite has a continually updated position output. These outputs are used to simulate the readings from the chasers' onboard range and location sensors. Each chaser uses its simulated sensor readings in its formation GNC system to update its orbit.

C. Formation GNC System

Each formation GNC system uses its own location in the ECI frame and the relative locations of the other satellites as inputs to calculate the direction of its desired location. It uses these inputs to generate a two-dimensional "potential field." An example potential field representing a single satellite is shown in Figure 1. The location of each other satellite is modeled as an area of low potential, or valley, with an area of high potential ringing each valley[2]. Negative values are shown as zero on these plots for scaling purposes so the potential field can be visualized better.

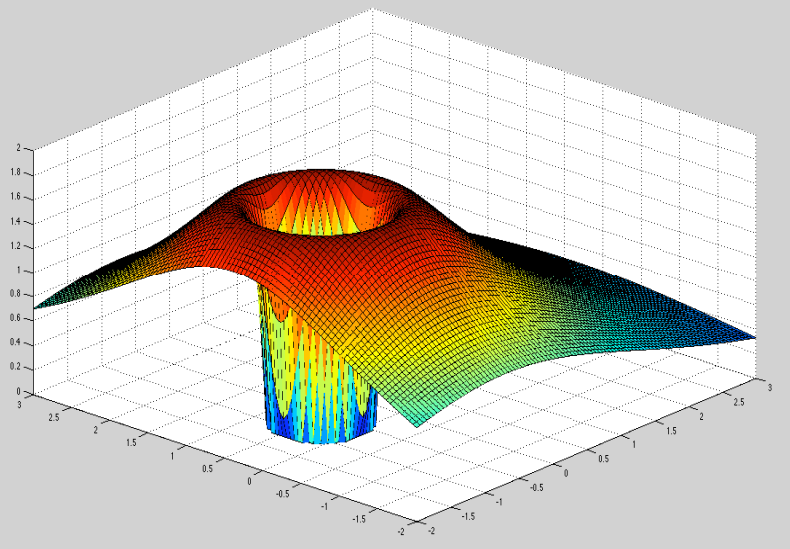

Figure 1. Potential field of one satellite

The value of r at each location of the potential field is

$$r = \text{sqrt}(x^2 + y^2) \qquad (1)$$

Equation 2 is the inverse polynomial equation used to model the valley at each satellite location.

$$q\_avoid = -a/r^6 \qquad (2)$$

Equation 3 is the decaying exponential used to model the areas of high potential.

$$q\_target = 6e^{(-.5r)} \qquad (3)$$

The constants in these equations were chosen so that the peak of the sum of these two equations would be a minimum of 0.2 km away from the center of the satellite.

The net potential field is calculated by adding the potential fields of each other satellite, shown in Figure 2. The desired location of the chaser vehicle is the top of the highest potential peak. In Figure 2 the initial location of the chaser vehicle is represented by the green square and the final target location is represented by the red square. The potential field function only uses satellite locations relative to the chaser, so the chaser location on this map is always at the origin. This potential field will bring all the satellites as close to each other as possible without letting them run into each other. For a formation with one lead satellite and two chasers this will result in a triangular formation with equal spacing between each unit.

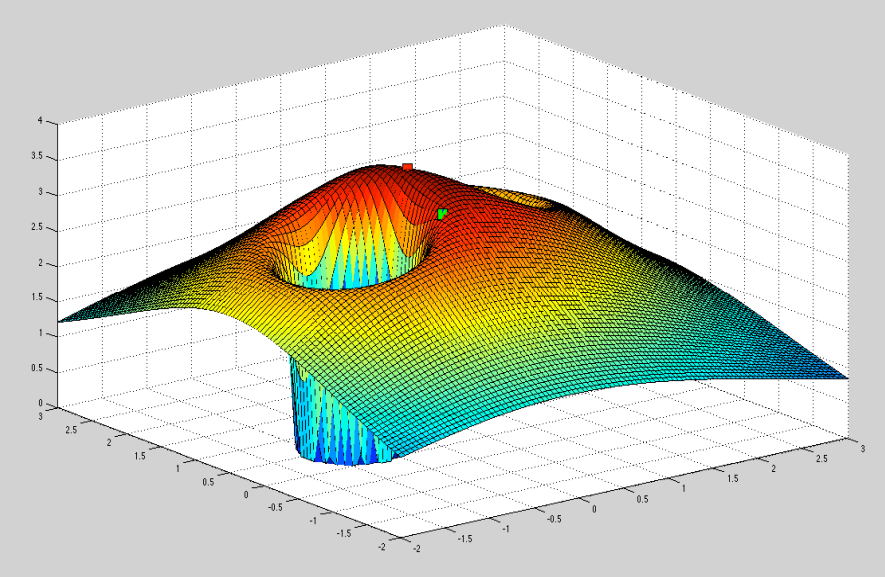

Figure 2. Net potential field, chaser and two other satellites

The potential field is used to find the direction the chaser needs to go to reach its final target location while avoiding the other spacecraft. This direction is calculated by solving for the gradient of the potential field at the location of the chaser. Using the gradient vector as the intermediate target location rather than the final target location avoids the possibility of the chaser running into another satellite on its way to the final target location. It has the added benefit that it is longer where the potential field is steepest, meaning that the chaser will navigate quickly away from the steep valleys surrounding the other satellites and it will move more slowly as it approaches its final target location.

The deltaV function calculates the velocity of the intermediate target location by approximating it as a spacecraft in a circular orbit using Equation 4, where v is velocity, mu is Earth's gravitational constant, and r is the radius of the intermediate target location in ECI coordinates.

$$V = \mathrm{sqrt}(mu/r) \qquad (4)$$

The relative position and velocity of the chaser vehicle to the intermediate target location are transformed from chaser-relative ECI coordinates to the CW coordinate frame, which is centered at the target location with x pointing radially outward, y pointing in the direction of motion perpendicular to x, and z perpendicular to both x and y. In

Figure 3 the ECI frame is red and the CW frame is green. The chaser and intermediate target locations are in the ECI frame and shown in black and the relative position is shown in blue.

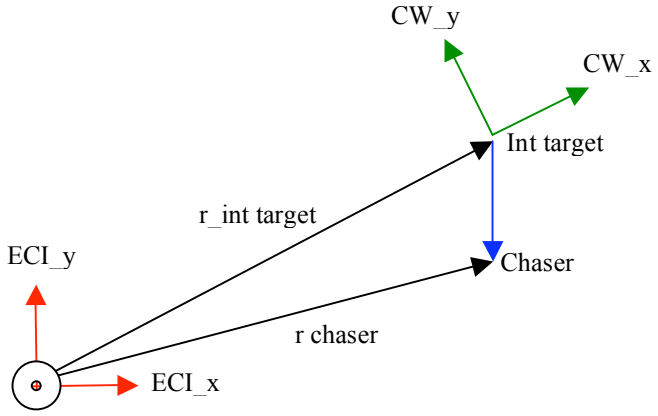

Figure 3. Chaser location and intermediate target location, ECI and CW coordinate frames

The deltaV function uses the CW equations to calculate the delta v needed for the chaser to reach the intermediate target location in the time specified, referred to as t_f. The delta v is recalculated every 30 seconds, which is less than t_f because the smaller the t_f, the higher the delta v must be to get the chaser to its position. The required delta v is recalculated before t_f has elapsed to avoid collisions, since other satellites are moving and changing their orbits as well. The delta v is converted back to ECI coordinates using the inverse of the CW transform and summed into the chaser's velocity to update the chaser dynamics.

D. Simulation Outputs

The output of the simulation is an animated graph showing the location of each satellite in ECI coordinates. Figure 4 shows a still shot of the animation from 2.5 minutes into the orbit. The blue arrow represents the lead satellite and the red and green ones represent the chasers. The initial coordinates of the three satellites are shown in Table 1.

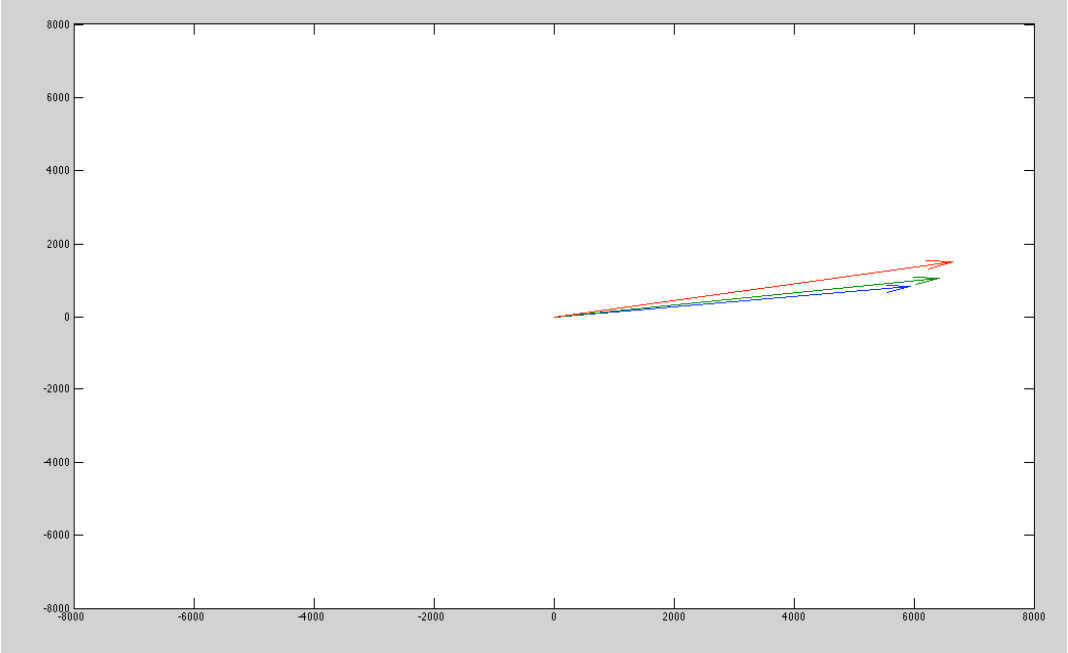

Figure 4. Location of satellites in ECI coordinate frame

Table 1. Initial satellite locations

Satellite	X location (km)	Y location (km)
Lead	6678	0
Chaser 1	6970	20
Chaser 2	6480	50

III. Conclusion

This GNC program fulfills its objectives of determining the location of the three satellites, determining the target locations of the chasers, using the CW equations to determine the needed delta v to reach those target locations, using a dynamics function to propagate the location of all the satellites, and generating a plot to demonstrate the results. It would be more robust if the equations used in determining the chasers' target locations were optimized, the time steps were made smaller, and the final time, t_f, assumed for calculating the delta V was optimized. Another possible problem with a system like this is that if one of the satellites besides the lead loses its ability to navigate, the rest of the formation would still follow it, even if it had a decaying orbit. The program could be easily updated to

work for any number of satellites, the 2D orbit could be expanded to a 3D orbit, and the shape of the formation could be changed by modifying the potential field function.

Appendix: Matlab Script and Sub-Functions

```
% Main Script: Senior Project Formation Flight Simulation, GNC
% All units are based on sec, km, rad
% Coordinates are in ECI unless otherwise specified
% Matlab functions:
% [t,a,v,r] = dynamics_a(t,r,rr,v,vv)
% dr0 = potential_field(dr1, dr2)
% [CW,CWtrans] = CW_frame(r,v)
% deltav = deltaV(dr0,dv0,n,tf)

% Define times (sec)
% Total time
T_total = 10000;
steps = floor(T_total/30);
step = 200;
% time for sat to reach target location, used to estimate delta v
tf = 60;

% initialize variable names
mu = 398600;
sum_deltav2 = 0;
sum_deltav3 = 0;
r1 = zeros(2,steps);  % Lead
r2 = r1;
r3 = r1;
v1 = r1;
v2 = r1;
v3 = r1;
r_rel12 = zeros(2,1);
r_rel32 = r_rel12;
r_rel13 = r_rel12;
r_rel23 = r_rel12;
dr2 = r_rel12;
dr3 = r_rel12;
dr2_ECI = r_rel12;
dr3_ECI = r_rel12;
dv2_ECI = r_rel12;
dv3_ECI = r_rel12;
dv2 = r_rel12;
dv3 = r_rel12;
drCW2 = r_rel12;
dvCW2 = r_rel12;
drCW3 = r_rel12;
dvCW3 = r_rel12;
deltav2 = r_rel12;
deltav3 = r_rel12;
vfinal2 = r_rel12;
vfinal3 = r_rel12;

% Initial positions
r1(:,1) = [6678 0]';  % Lead
r2(:,1) = [6970 20]';
r3(:,1) = [6480 50]';
```

```
% Unit normal
u1 = [r1(2,1)/norm(r1(:,1)), r1(1,1)/norm(r1(:,1))];
u2 = [r2(2,1)/norm(r2(:,1)), r2(1,1)/norm(r2(:,1))];
u3 = [r3(2,1)/norm(r3(:,1)), r3(1,1)/norm(r3(:,1))];

v1(:,1) = sqrt(398600/norm(r1(:,1)))*u1;
v2(:,1) = sqrt(398600/norm(r2(:,1)))*u2;
v3(:,1) = sqrt(398600/norm(r3(:,1)))*u3;

for k = 1:5 %(steps-1)
    % determine rel_r for #2 & #3
    r_rel12(:,1) = r1(:,k) - r2(:,k);
    r_rel32(:,1) = r3(:,k) - r2(:,k);

    r_rel13(:,1) = r1(:,k) - r3(:,k);
    r_rel23(:,1) = r2(:,k) - r3(:,k);

    % rel pos of target location
    dr2(:,1) = potential_field(r_rel12(:,1), r_rel32(:,1),step);
    dr3(:,1) = potential_field(r_rel13(:,1), r_rel23(:,1),step);

    % abs pos of target location
    dr2_ECI(:,1) = dr2(:,1) + r2(:,1);
    dr3_ECI(:,1) = dr3(:,1) + r3(:,1);

    % abs vel of target location
    dv2_ECI(:,1) = sqrt(mu/norm(dr2_ECI(:,1)));
    dv3_ECI(:,1) = sqrt(mu/norm(dr3_ECI(:,1)));

    % rel vel of target location
    dv2(:,1) = dv2_ECI(:,1) - v2(:,k);
    dv3(:,1) = dv3_ECI(:,1) - v3(:,k);

    % Define n
    n2 = norm(v2(:,k))/norm(r2(:,k));
    n3 = norm(v3(:,k))/norm(r3(:,k));

    % Change coords ECI -> CW, find CW matrix, change rel vectors from
    % chaser origin to target location origin, then transform
    [CW2,CWtrans2] = CW_frame(dr2(:,1),dv2(:,1));
    drCW2(:,1) = CW2*(-dr2(:,1));
    dvCW2(:,1) = CW2*(-dv2(:,1));

    [CW3,CWtrans3] = CW_frame(dr3(:,1),dv3(:,1));
    drCW3(:,1) = CW3*(-dr3(:,1));
    dvCW3(:,1) = CW3*(-dv3(:,1));

    % determine delta v for #2 & #3
    deltav2(:,1) = deltaV(drCW2(:,1), dvCW2(:,1), n2,tf);
    deltav3(:,1) = deltaV(drCW3(:,1), dvCW3(:,1), n3,tf);

    % total the delta v values for later to track efficiency
    sum_deltav2 = sum_deltav2 + norm(deltav2(:,1));
    sum_deltav3 = sum_deltav3 + norm(deltav3(:,1));
```

```
    % Transform back to ECI frame, these values are still rel to
chaser
    vfinal2(:,1) = CWtrans2*deltav2(:,1);
    vfinal3(:,1) = CWtrans3*deltav3(:,1);

    % update velocity, then feed into dynamics sim
    v2(:,k) = v2(:,k) + vfinal2(:,1);
    v3(:,k) = v3(:,k) + vfinal3(:,1);

%     Dynamics function call
%     Input is r1(:,k), v1(:,k), etc.
%     Output is r1(:,(k+1)), v1(:,(k+1)), etc. for 30 seconds later
for t = 1:30
    [v1(:,(k+1)),r1(:,(k+1))]=
dynamics_a(t,r1(1,k),r1(2,k),v1(1,k),v1(2,k));
    [v2(:,(k+1)),r2(:,(k+1))]=
dynamics_a(t,r2(1,k),r2(2,k),v2(1,k),v2(2,k));
    [v3(:,(k+1)),r3(:,(k+1))]=
dynamics_a(t,r3(1,k),r3(2,k),v3(1,k),v3(2,k));
end
    % plot separately dr2 & dr3 (rel pos of target location)
%     figure(1)
%     hold off
%     quiver(dr2(1,k),dr2(2,k))
%     figure(2)
%     quiver(dr3(1,k),dr3(2,k))
%     % plot separately drCW2 & drCW3 (same thing except in CW coords)
%     figure(3)
%     quiver(drCW2(1,k),drCW2(2,k))
%     figure(4)
%     quiver(dvCW2(1,k),dvCW2(2,k))
    % plot r1,r2,r3
    figure(5)
    hold off
    quiver(r1(1,k),r1(2,k))
    hold on
    quiver(r2(1,k),r2(2,k))
    quiver(r3(1,k),r3(2,k))
    axis([-8000 8000 -8000 8000])
    hold off
    % pause(.001)  % time in sec
end
```

Sub-Funtions

```
function [CW,CWtrans] = CW_frame(r,v)
% This function uses the target location and velocity
% to generate the CW unit frame
% Output is two 2x2 matrices
CW = [r/norm(r), v/norm(v)];
CWtrans = CW';
% transformation matrix, same as CW, spelled out in detail (3D)
% Q =
% [r(1)/norm(r),r(2)/norm(r),r(3)/norm(r);...
% v(1)/norm(v),v(2)/norm(v),v(3)/norm(v);...
% khat(1),khat(2),khat(3)];
end
```

```
function deltav = deltaV(dr0,dv0,n,tf)
% Use CW eqns to find deltaV1, dv1
% deltaV1 is impulse needed
% dv1 is rel velocity after impulse in CW frame, doesn't actually
% matter because delta v is recalculated in 30 sec, long before tf arrives
tol = .005;  % distance in km
a = n*tf;

PHIrr = [(4-3*cos(a)),0;6*(sin(a)-a),1];
PHIrv = (1/n)*[sin(a),(2-2*cos(a));(2*cos(a)-2),(4*sin(a)-3*a)];
PHIvr = n*[3*sin(a),0;(6*cos(a)-6),0];
PHIvv = [cos(a),2*sin(a);-2*sin(a),(4*cos(a)-3)];

if norm(dr0)<tol          % If the chaser is within tol distance of target
loc, delta v to stop
    % delta v if rel distance < tolerance
    deltav = -(PHIvr - PHIvv*(PHIrv^-1)*PHIrr)*dr0;
    % rel velocity is zero after the impulse deltav1
    % dv1 = [0 0]';
else
    % dv1 = actual rel velocity after impulse 1
    dv1 = -(PHIrv^-1)*PHIrr*dr0;
    % deltav1 = deltaV of impulse
    deltav = dv1 - dv0;
end
```

```
function [v,r] = dynamics_a(t,r,rr,v,vv)
r0 = [r,rr];
v0 = [v,vv];

a = t*0-398600*r0/(norm(r0)^3);
v = a*t+v0;
r = v*t+r0;

a = a';
v = v';
r = r';
end
```

```
function qtotal_value = potential(x,y,dr1,dr2,r)
x1=x-dr1(1);
y1=y-dr1(2);
x2=x-dr2(1);
y2=y-dr2(2);
a = (4-r^2)*r^6;

% Area avoided (location of first S/C)
q1=-a/((x1^2)+(y1^2))^3;
% Target location rel to first S/C
q1t=3*exp(-.5*sqrt((x1^2)+(y1^2)));

% Area avoided (location of second S/C)
q2=-a/((x2^2)+(y2^2))^3;
% Target location rel to second S/C
q2t=3*exp(-.5*sqrt((x2^2)+(y2^2)));

qtotal_value = q1 + q1t + q2 + q2t;

% Use this part to generate better plots, ignore it when running the
actual program
% if qtotal_value<=0
%     qtotal_value = 0;
end
```

```
function dr0 = potential_field(dr1, dr2)
% Goal: reach point of highest potential

% r = radius of area to avoid (km)
r = .5;

% specifies step size and min/max dims for meshgrid
% so that 0,0 will always be a location in the meshgrid
step = 100;
mesh_min = floor(min(cat(1,dr1,dr2,0)));
mesh_max = ceil(max(cat(1,dr1,dr2,0)));
mesh_min = (fix(mesh_min/step)-1)*step;
mesh_max = (fix(mesh_max/step)+1)*step;

% step size such that: step*(whole number)=1
[x,y]=meshgrid(mesh_min:step:mesh_max);

% Define variables
m = length(x);
n = length(y);
qtotal = zeros(m,n);
qstore = 0;
target_location = [0,0];
chaser_location = [0,0];
dr0 = [0,0]';
chaser_direction = zeros(m,n);

for i=1:m
```

References

[1]Curtis, Howard D: "Orbital Mechanics for Engineering Students," Elsevier Butterworth-Heinemann, 2005

[2]Langelaan, Jack and Rock, Steve: "Navigation of Small UAVs Operating in Forests," *AIAA Guidance, Navigation and Control Conference*, Providence, Rhode Island, 2004.

[3]Leitner, Jesse; Bauer, Frank; How, Jonathan; Moreau, Michael; Carpenter, Russell and Folta, David: "Formation Flight in Space: Distributed Spacecraft Systems Develop New GPS Capabilities." GPS World. Feb 1, 2002. http://www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=9518