Development and Design of a Hovering Controller for Operation in a Dynamic Asteroid
Environment

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

Nicholas Georgiades

June, 2011

Development and Design of a Hovering Controller for Operation in a Dynamic Asteroid Environment

Nicholas Georgiades[*]

California Polytechnic University, San Luis Obispo, CA, 93405

The project seeks to develop a dynamic model similar to that present near a solar system small body, and to design a controller suggested in other resources that will allow a spacecraft operating in this environment to hover in a fixed location in the relative reference frame of the small body. The paper discusses the derivation of the equations of motion used in the non-linear dynamic model, the design of the controller that will allow the spacecraft to hover, and the development of the control loop that will simulate the spacecraft hovering in the dynamic environment of the asteroid 1999 JU3. A phase plane analysis is conducted to validate the stability of the system, as well as provide a means for adjusting the accuracy to which the controller hovers about the desired set point. The project is successful in all the stated goals, and improvements to the project are discussed.

I.   Introduction

As unmanned space exploration expands to include asteroids and comets in the list of possible destinations, it becomes vital to find new ways to stably remain in orbit about these small bodies. One of the proposed methods for remaining stable about these small bodies is hovering. Hovering has an advantage over more conventional orbits in that they can better handle the unpredictable and varying nature of a small bodies' dynamic environment[1,3,4].

This project seeks to develop a control algorithm and a controller that will enable a spacecraft to hover in a fixed position between the Sun and a small solar system body. The challenges in developing this controller will be in establishing the dynamic environment of the small body, and validating that the controller works effectively in a non-linear dynamic model. Overall the project can be broken down into three phases: research, development, and validation.

Before work began on the controller, research on the topic was conducted, however there will be no discussion of the details of these resources. They have been included in the references section of this paper, and any reference to them within the paper will be noted with the appropriate source. Next some of the methods and techniques discussed in the papers were implemented and tested in a dynamic model that I assembled in MATLAB and SIMULINK. This paper seeks to discuss the development, testing, and validation of the dynamic model and the controller that operates therein.

II.   Governing Equations of Motion

A. Primary Body's Gravitational Effects

The basic dynamics of the system that the spacecraft will need to operate in are governed by gravitational effects from asteroid so long as the spacecraft is positioned well within the Hill Sphere of the body. The radius of the Hill Sphere can be calculated using the equation

$$r_H = a(1-e)^3 \sqrt[3]{\frac{\mu_{JU3}}{3\mu_{sun}}} \tag{1}$$

The Hill Sphere of the asteroid 1999 JU3, with a semi-major axis, a, of 1.7788e8 km and an eccentricity, e, of 0.19, and a standard gravitational parameter, μ_{JU3}, of approximately 4.28e-8 $\frac{km^3}{s^2}$, is approximately 68.5 km. Of course, the actual effectiveness of the asteroid's gravity within this sphere will vary depending on the proximity of the asteroid to the sun throughout the period of its orbit. For the purposes of this project, a position well within the

---

[*] Student, Aerospace Engineering Department, 1 Grand Ave

asteroid's Hill Sphere was selected to ensure that the asteroid's gravitational pull was the dominant force. With this assumption made, we can use Newton's universal law of gravitation with Newton's second law of motion as given in the equation

$$\vec{a}_g = -\frac{\mu_{JU3}}{r^3}\vec{r} \tag{2}$$

where $\vec{a}_g$ is the acceleration due to the primary body's gravity and $\vec{r}$ is the position vector of the orbiting secondary body, to describe the primary motion of the satellite in the asteroid's frame of reference. The small nature of the primary body suggests that gravitational effects alone will not be the predominating terms in the forcing function. Therefore, in an effort to ensure that the controller will be robust, a more accurate environment model was incorporated, taking into account additional perturbing forces, namely third body gravitational effects, solar radiation pressure, and a non-uniform primary gravity field, that will be discussed in the following sections.

B. Third Body Effects

The next term of the forcing function that will be discussed are the effects that a third body will have on the dynamics of the operating environment. The only third body that was used in the model was the Sun, since both the spacecraft and the asteroid it is hovering above will be completely subject to its gravitational pull. In an effort to more accurately model the effects over time, the dynamic model incorporated both the asteroid's and the satellites motion about the Sun. Simply put, this motion can be described using Eq. 2, substituting the Sun's standard gravitational parameter, μ_{sun}, for the asteroid's and using the asteroid's and spacecraft's position where appropriate.

In the model however, the system needed to propagate the absolute position of the asteroid and the spacecraft together first, before examining the gravitational effect the asteroid had on the spacecraft. As a result, the state vector for the equations of motion is given in terms of the absolute position and velocity of the asteroid and the spacecraft, rather than the spacecrafts position and velocity relative to the asteroid. For this we substitute the absolute positions of the spacecraft and the asteroid, defined in the model as $\vec{r}_1$ and $\vec{r}_2$ respectively, in for the value of $\vec{r}$ in Eq. 2. This yields a new equation for the primary body's acceleration due to gravity[4]

$$\vec{a}_{g1} = -\frac{\mu_{JU3}}{(r_1 - r_2)^3}(\vec{r}_1 - \vec{r}_2) \tag{3}$$

leaving the acceleration due to third body effects from the Sun in the form of Eq. 2. Together the acceleration due to gravitational effects in terms of the inertial position of the spacecraft and the asteroid can be described by the equation[4]

$$\vec{a}_{G1} = \vec{a}_{g1} + \vec{a}_{3b1} = \frac{\mu_{JU3}}{(r_2 - r_1)^3}(\vec{r}_2 - \vec{r}_1) - \frac{\mu_{Sun}}{r_1{}^3}\vec{r}_1. \tag{4}$$

A similar system of equations exists for the asteroid's motion, however since the mass of the spacecraft is so insignificant when compared to the mass of the asteroid, it is assumed that the spacecraft's gravitational force exerted on the asteroid is negligible. As a result, the asteroid's acceleration due to gravity, needed to update the value of $\vec{r}_2$, can be expressed in the form of Eq. 2, rewritten as

$$\vec{a}_{G2} = \vec{a}_{g2} = -\frac{\mu_{Sun}}{r_2{}^3}\vec{r}_2. \tag{5}$$

With these two equations, the model in its most basic form is ready for analysis. The next two sections will describe two additional perturbations that were included in the model for the purpose of evaluating the robustness of the controller.

C. Solar Radiation Pressure

Although not usually considered a significant source of translational acceleration due to its categorization as a conservative force for conventional orbits, solar radiation pressure was included in this model because of the nature

of the anticipated trajectory. The controller is designed to keep a spacecraft in a fixed position with respect to the Sun and the asteroid. This means that solar radiation pressure can no longer be considered a conservative force, since it will always be acting in the same relative direction for the entire duration of the controller's operation. The acceleration due to solar radiation pressure can be determined form the equation[4]

$$\vec{a}_{srp1} = -\frac{C_S C_R A_{CS}}{c}\frac{\vec{r}_1}{r_1^3} \tag{6}$$

where C_S, is the solar constant $1367 \frac{W}{m^2}$, C_R is the assumed coefficient of reflectivity of the spacecraft of 1.5, A_{CS} is the assumed cross-sectional area of the sun facing side of the spacecraft of 3 m^2, and c is the speed of light, or $2.998e - 8 \frac{m}{s}$.

The inclusion of this specific term in the forcing function is optional depending on the location of the small body the spacecraft will hover around. For example, the asteroid examined in this report, JU3, is a Near Earth Asteroid, and its distance from the Sun ranges from 0.96 AU to 1.4 AU. This leads to a wide variation in the magnitude of the solar radiation pressure, since the value itself varies inversely as a function of the square of the distance from the Sun.

D. Non-Uniform Gravity Field (not implemented)

The biggest challenge in developing the environmental dynamic model was developing a means to incorporate a non-uniform gravity field. The earliest versions of the controller were tested assuming the central body to be a point mass rather than an actual volume of mass. Since the exact models of the asteroid's could not be found or developed, a randomized asteroid model was going to be developed to generate a volume to test the control algorithm with. This model would introduce the perturbing effects due to a non-uniform gravity field, and validate whether the controller is in fact robust.

Unfortunately, time limitations prevented the incorporation of the randomized asteroid model that was developed by the author. Instead, the controller was verified using only a point mass assumption, and the incorporation of a non-uniform gravity field will be left as a possible expansion for a future project.

E. Implementation of the Equations of Motion

Finally the full expression for the perturbing accelerations on the spacecraft in the asteroid's dynamic environment can be written. Using Eqs. 4, 5, 6, and 8, the equations of motion can be written as

$$\vec{a}_1 = \vec{a}_{G1} + \vec{a}_{srp1} \tag{7}$$

$$\vec{a}_2 = \vec{a}_{G2}. \tag{8}$$

These equations of motion were written in an embedded MATLAB function within the SIMULINK model and can be seen in the Appendix. To verify that the equations of motion were in fact working properly, the system was allowed to operate without an operating controller for an extended period of time. The output can be seen in Figure 1. The result is as expected, as the motion seen is exactly what the motion of a slowly orbiting object in three body system would look like[4]. The slight variation in the eccentricity and semi-major axis of the orbit, evident in Figure 2 can be attributed to the perturbations in the system.
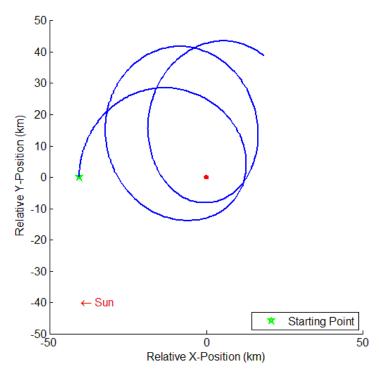
Figure 1. The uncontrolled motion of an object starting at 40 km altitude above the surface in the relative frame for 100 days looks like the motion of an object in a three body system.
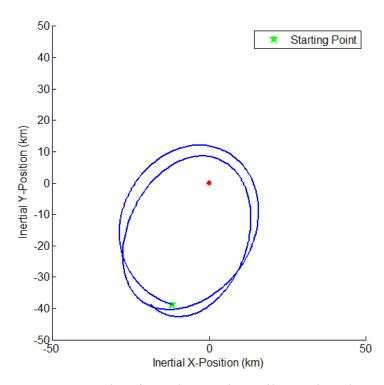

Figure 2. The same uncontrolled motion of an object starting at 40 km altitude in the inertial frame shows variances in the orbital elements due to third body effects and the changing value of the Hill Sphere.

This motion is exactly what the controller needed to be designed to counteract, effectively creating an equilibrium point at a fixed point in the relative frame.

III.   Controller Design

A. The Plant

The equations of motion covered in the previous section are set up in such a way that the states are propagated in the inertial frame with the Sun as the primary central body. This is not suitable for this controller, as the design intends to utilize the relative position as the primary indicator of whether or not a correction needs to be made. This meant that a series of coordinate transformations needed to be included within the plant to convert the state output from the equations of motion from the inertial frame into the relative frame. Then decisions whether or not to implement control would be made with information supplied in terms of the relative frame, and control added in the same manner. Before the control input could be fed back into the equations of motion, the input had to be converted from an acceleration in the relative frame into an acceleration in the inertial frame. Figure below shows exactly this process with the necessary plant inputs and outputs.
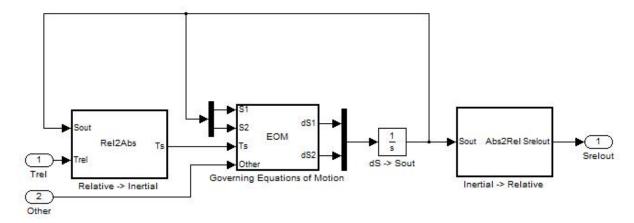

Figure 3. The plant required a transformation from the inertial frame that the state is propagated in, to the relative frame in which the control is applied.

B. Controller

1. The Schmitt Trigger

The main mode of control used is the Schmitt trigger. The Schmitt trigger, composed of two relays, establishes a dead band on the control region. Dead bands are necessary in implementing hovering control because without them there is the potential for too much energy to be added to the system[2].

Schmitt triggers, like the one shown in Figure 4 were placed in the system to turn on or off based on the spacecraft's distance from the point mass. A relay is used to check the relative position and will enable if the position is outside of the set point in either direction.
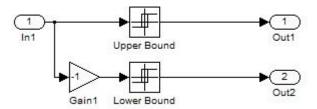

Figure 4. A Schmitt trigger was used to control the relative hovering position.

These first attempts at open loop hovering drove the system to instability. The control pulses would have to be ridiculously small in order for open loop hovering control to be effective, or the system would need to use continuous correction with a very low gain. This controller is intended to work with a higher gain non-continuous system, and can be thought of utilizing small thrust attitude control system thrusters to hover. As a result, control only on the relative position is not enough.

2. Pulse-Width Pulse-Frequency Modulation

In order to establish a tighter control on the position of the hovering spacecraft, a form of pulse-width and pulse-frequency (PWPF) modulation needs to be implemented to prevent the stabilizing thrusters from adding too much energy to the system. I implemented a form of PWPF modulation by introducing a second tier of Schmitt triggers on the relative velocity of the hovering spacecraft with respect to the small body. This acts as a mode for controlling pulse-width of the thruster firings, while the combination of the first and second tier of Schmitt triggers controls the frequency of these pulses.

Figure 5 shows the second tier of Schmitt triggers controlling the components of the relative velocity. These triggers act as a go or no go check for the thruster firing. If the spacecraft velocity is higher than a value specified by the user, then the second tier of triggers will disable the thruster, even if the sensor says that the relative position is outside of the dead band specified by the first tier. The same thing could be done in only the opposite direction of gravitational acceleration, as this is the primary thrust direction, with a single relay, however I wanted to ensure that all three axes would be rate limited.
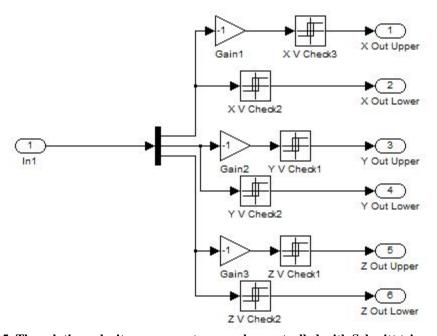

Figure 5. The relative velocity components were also controlled with Schmitt triggers to ensure that there would be a small range of acceptable velocities at which the controller would not be disabled.

3. Position and Velocity Control Interface

An example interface between the two tiers of Schmitt triggers for each of the components can be seen in Figure 6. In the figure below, the X Thrust Trigger block is the Schmitt trigger shown previously in Figure 4. An additional level of manual control has been added in the form of an on/off switch so that control in specific directions can be completely disabled in a higher subsystem.
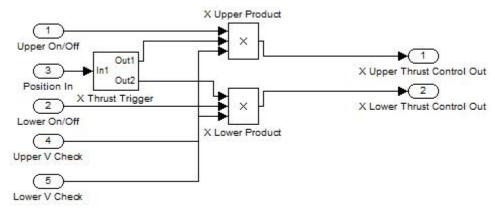
Figure 6. The interface between the second tier of Schmitt Triggers, input through Ports 4 and 5 can be seen for the x axis thrust control.

4. Assembled Thrust Controller

A total of three of the interfaces shown in Figure 6 were used in parallel to control the x, y, and z axis thrusters in the relative frame. Combining the three interfaces and the velocity controller shown in Figure 6, we arriving at the basic controller used to force the spacecraft to hover in the relative frame at the desired position. In Figure 7 we can see the assembled controller with all of the inputs, outputs, and the implemented thrust control direction switches. It may be difficult to see, but the magnitude of the thrust (expressed as a specific thrust here in terms of acceleration) is not added into the system until after the control state of every component is already determined. The outputs of each of the Schmitt triggers within the system only need to output a one or zero as a result. Setting the correct set points for the dead bands on each of these Schmitt triggers will be discussed in a later section.
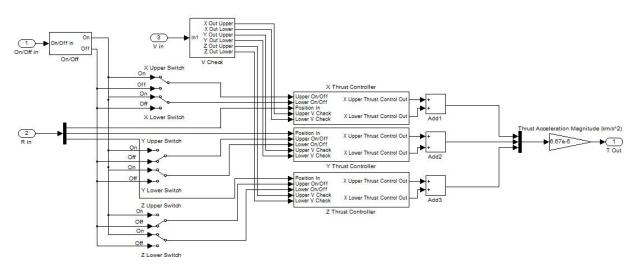

Figure 7. The thrust controller requires realtive position and velocity knowledge, but gives the user the option of disabling control in any specific direction.

This completes the discussion of the controller design. The only remaining piece of the control loop to address is the way in which the controller is supposed to interface with sensors located on the spacecraft.

C. Sensor and System Interface

In order to use the controller onboard, the system will need to interface with sensors that can actually provide the controller with the information needed to accurately control hovering. Since this is designed to be a closed loop system, the error between the desired hovering location and the actual spacecraft relative position with respect to the small body also needs to be found. Both of these concerns are assumed to be taken care of in the implementation of

this controller, but sensor compatibility is a serious concern when deciding how tight of a dead band the hovering controller is designed to have.

For this controller, it is assumed that the sensors onboard the spacecraft are capable of providing the command and data handling system (C&DH) with the relative position and velocity knowledge required. With this assumption a simple difference between the state output in the relative frame and the desired relative position is all that is used, and can be thought of as being supplied in this format by the C&DH system.

D. The Control Loop

With the plant, controller, and sensor interface established, we can see the assembled control loop below in Figure 8 There is a delay incorporated so that the spacecraft may fall towards the asteroid initially before the system activates and corrects the spacecrafts position.
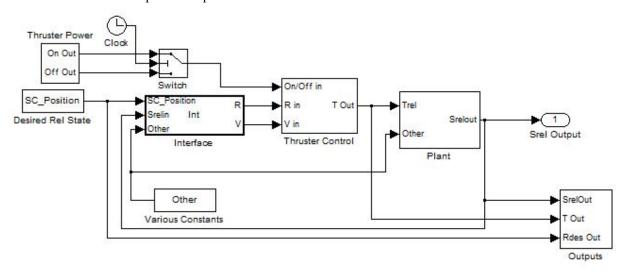
Figure 8. The control loop incorporates the plant, the thruster controller, and the system interface and outputs the necessary information for analysis.

IV.   Results

The controller was tested using a number of constant parameters. The small body used was the asteroid 1999 JU3, which has the orbital elements shown in Table 1 for the date May 4, 2018. The radius of the Hill Sphere for this asteroid is approximately 68.5 km, so a point well within this region is required to minimize the effect of the Sun's gravitational pull. An altitude of 4 km was selected as the operating position for its position well within the Hill Sphere and its distance of nearly 4 radii from the asteroid's surface.

Table 1. The physical properties and orbital elements for the asteroid 1999 JU3 can be seen below.

Small Body	*Asteroid 1999 JU3*	
Physical Properties		
Mean Radius	R	0.550 km
Mass	M	6.4143×10^{11} kg
Standard Gravitational Parameter	μ	$4.27 \times 10^{-8} \frac{km^3}{s^2}$
Classical Orbital Elements		
Semi-Major Axis	a	1.778×10^8 km
Eccentricity	e	0.19
Inclination	i	5.88°
Right Ascension of the Ascending Node	Ω	251.7°
Argument of Perigee	ω	211.3°
Period	T	473.6 days

The controller was run for a period of one hundred days with a desired altitude of 4 km, and asteroid sun angle of 180°, putting the hovering location between the Sun and the asteroid in the relative frame. The result can be seen below in Figure 9. The lack of a trajectory in the relative frame is an indication that hovering has been achieved, as the spacecraft remains in the same approximate position for the duration of the simulation. The motion in the inertial frame can clearly be seen in Figure 10. where the spacecraft completes one period of it's very slow orbit about the asteroid in the amount of time it takes the asteroid to orbit the Sun. Since the simulation was only run for 100 days, the spacecraft has only completed a portion of its 473 day orbit.
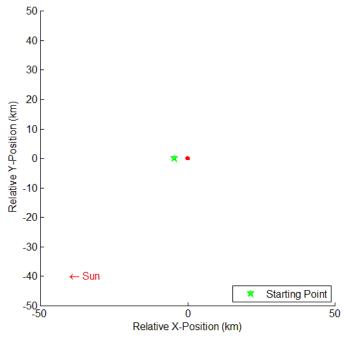
Figure 9. The hovering controller appears to work as the motion in the relative frame has been stopped.
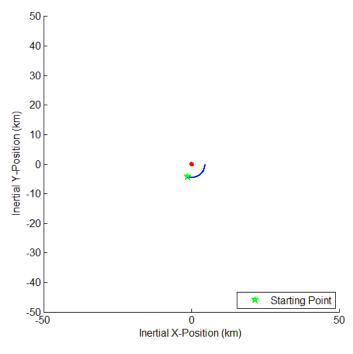

Figure 10. The spacecraft appears to be in a an orbit with a period that matches that of the asteroid's period about the Sun.

10

The motion with the controller disabled in Figure 11 is a stark contrast with the motion seen when the controller is turned on as seen in Figure 9. Here, in the relative frame, the spacecraft completes several full orbits about the asteroid with a few very small perturbations visible due to the Sun's gravitational pull.
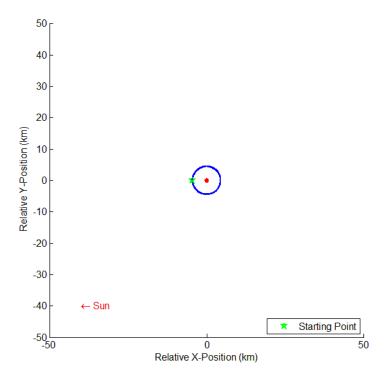

Figure 11. The uncontrolled relative motion of the spacecraft beginning at an altitude of 4 km.

Now let's take a closer look at what is actually happening in the relative frame when the controller is enabled. If we look at Figure 12, we can confirm that the desired position and system output are virtually identical. If we look at the right side of the figure, we can see very small fluctuations in the y-component of the relative position. This is due to a less stringent dead band in the y-direction. This was intentional, since for the position chosen, the primary acceleration direction will be in the positive x-direction, the direction in which the asteroid is relative to the spacecraft.

If you look back up in the controller design section at Figure 7, you can actually see that the control in the positive y-direction has been disabled. The dynamics of the system will naturally stabilize the motion of the spacecraft, so a thruster firing in that direction is not necessary for stability. The same phenomenon is true for the z-axis as well, control is disabled in the stabilized direction. If a fixed position is not a concern, and only the altitude of the spacecraft needs to be maintained, then the off axes directions can actually be completely disabled, and the spacecraft will naturally drift about in the stability region[7], oscillating indefinitely about the equilibrium point at zero for both the y and z axes while control will continue to exist in the positive x direction.
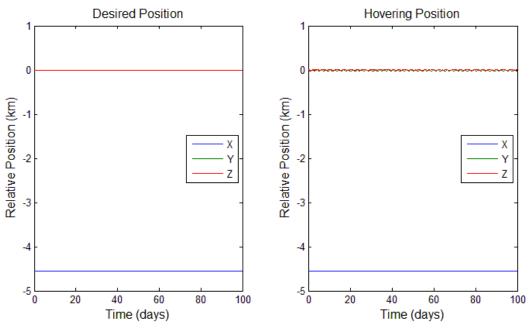
Figure 12. The actual hovering position shown on the right closely matches the desired hovering position shown on the left.

Now let's shorten the duration of the simulation, since we have verified that it is effective at supporting extended periods of hovering. If we shorten the simulation time to one day, we can see the motion that occurs in the x direction in Figure 13. The y and z outputs are not shown so that we can focus on just the primary control direction.
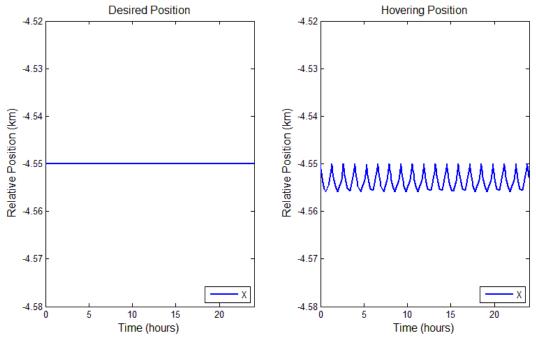

Figure 13. The activity of the controller is plainly visible in the x-direction when the simulation time is only one day.

The controller appears to be effective, and can be adjusted to exercise more or less control as the user desires.

V.   Analysis

The controller appears to be working effectively, and stabilizing the spacecraft's motion in the relative frame. In this section we will take a look at the phase plane of the state vector to verify that the motion is in fact stable. Upon immediate inspection, the signature parabolic shape of an oscillation about a dead band can be seen in the phase plane[6].

If we examine the phase plane of the state vector for a duration of one day, we see the graph in Figure 14. Here the simulation has only been run for 79 minutes and 15 seconds to get a better resolution on the phase plane. At this point, nearly one cycle in the hovering oscillatory motion has been completed, and we can get an idea of the stability of the controller. The starting location is plainly visible, and immediately a thruster fires accelerating the spacecraft from a relative velocity of virtually zero to a relative velocity of a $0.5 \frac{cm}{s}$ away from the asteroid. The limit of $0.5\frac{cm}{s}$ was set as the velocity limit in the second tier of Schmitt triggers. This limit translates to an oscillation in the position of the spacecraft of 6 m.
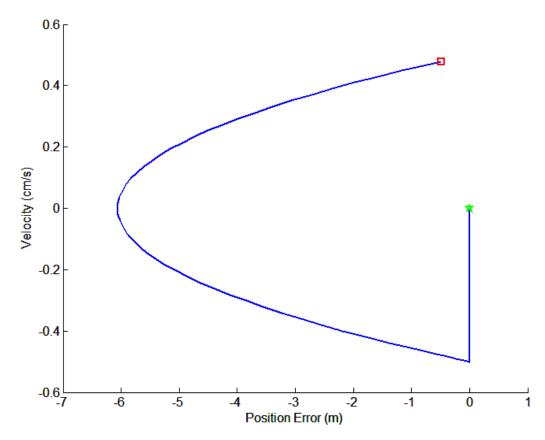

Figure 14. The phase plane diagram for a position dead band of 6 m and a velocity dead band of $1\frac{cm}{s}$.

Immediately a relationship can be developed between the limit posed on the duration of the thruster firings. The parabola shown in the phase plane gives a baseline to develop other velocity limits depending on the size of the position dead band. For example, assume we want the dead band to be increased to 20 m. If we flip the axes of the plot, the phase plane can be approximated by the equation of a parabola

$$x = ay^2 + b \tag{9}$$

since we know the roots to be roughly $\pm0.5\frac{cm}{s}$ and the x-intercept to be -6 m, we can solve for the coefficient a. Solving for a, we get a value of $a = 24$. Next we write the new equation for the phase plane

$$x = 24y^2 - 20. \tag{10}$$

Solving for the roots of this new equation gives the values of the new velocity limits, $y = \pm 0.912 \frac{cm}{s}$. When the dead band on the velocity values was adjusted to the new values, and the duration of the simulation extended to 150 minutes, the phase plane in Figure 15 was obtained.
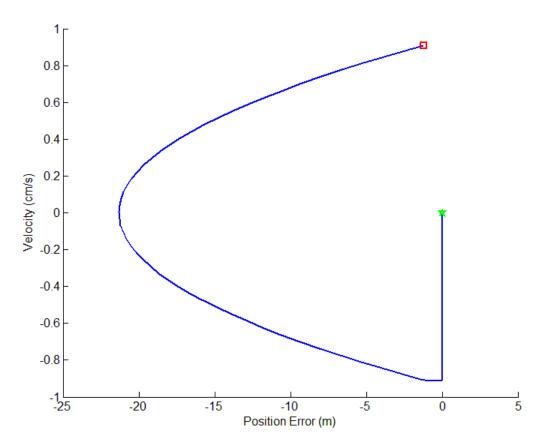

Figure 15. The phase plane for a dead band of 20 m was achieved by adjusting the parabola fitted to the 6 m dead band velocity limits.

The phase plane analysis has both provided a means of verifying the stability of the controller as well as a means for establishing reasonable values for the velocity limits by specifying a desired position dead band.

VI.   Conclusion

This project sought to develop a controller that was capable of allowing a spacecraft to hover between a solar system small body and the Sun, and the results indicated that the attempt was successful. The phase plane analysis indicates that the controller induces stable harmonic motion about a set point, and the plots of the motion in the relative reference frame indicate that the spacecraft does indeed hover in a fixed location with respect to the Sun and the asteroid 1999 JU3.

Although the project is a success, there are a number of improvements that could certainly be made to the controller. The first, as mentioned earlier in the report, is the incorporation of a non-uniform gravity field model into the equations of motion to verify that the controller can handle additional perturbations in the system and that the controller is robust. Additional tests could be made with the controller, verifying that it works at several locations within the Hill Sphere of the asteroid. It is during these tests that the limitations of the controller become apparent. In hindsight, the controller was developed with a serious shortcoming. All of the controller's components are based in the small body's relative frame. A drastic improvement to this implementation would be to switch the controller to operate in the spacecraft's body frame, and have an additional coordinate transformation from the body frame to the

14

relative frame of the asteroid. This would require to have additional knowledge of the spacecraft's attitude in the relative reference, not simply it's relative position. One final improvement that could go with this change is the incorporation of an attitude controller to constantly adjust the spacecraft's attitude with respect to the asteroid and Sun, keeping the same faces of the spacecraft always facing the same bodies at all points during its orbit. This project merely assumed that such a controller already existed and operated within the system.

Despite these limitations, the controller at the very least proves that the control algorithm for hovering in a fixed location in the relative reference frame is theoretically possible, and that using dead band control is a valid means of establishing a hovering trajectory.

Appendix

A. Source Code for the Embedded MATLAB Functions within the Control Loop

The Equations of Motion Block:

```matlab
function [dS1,dS2] =  EOM(S1,S2,Ts,Other)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

%CONSTANTS
Sun.mu = Other(1,1);   %1.32712e11 km^3/s^2
JU3.mu = Other(2,1);   %4.28e-8    km^3/s^2
Fsrp   = Other(3,1);   %Solar Radiation Pressure Constant

%STATE INPUT
r1 = S1(1:3,1);        %Spacecraft Position State [x;y;z]km
v1 = S1(4:6,1);        %Spacecraft Velocity State [x;y;z]km

r2 = S2(1:3,1);        %Asteroid Position State   [x;y;z]km
v2 = S2(4:6,1);        %Asteroid Velocity State   [x;y;z]km

%EQUATIONS OF MOTION
%Acceleration terms currently incorporate:
%   Perturbations due to Solar 3rd Body Effects
%   Perturbations due to Solar Radiation Pressure
%Need to incorporate:
%   Perturbations due to Non-Spherical Body/Non-uniform Gravity Field
a3b  = JU3.mu*(r2-r1)/(norm(r2-r1))^3;    %3rd Body Effects
asrp = Fsrp*r1/norm(r1)^3;                %Solar Radiation Pressure
ansb = 0;                                 %Non-spherical Body
ap   = a3b+asrp+ansb;                     %Total Perturbations

a1   = -Sun.mu*(r1)/(norm(r1))^3+ap+Ts;   %Spacecraft acceleration equation
a2   = -Sun.mu*(r2)/(norm(r2))^3;         %Asteroid acceleration equation

%CONVERSION FROM ABSOLUTE TO RELATIVE (separate block atm)
% Rrel  = r1-r2;
% OM    = cross(r2,v2)/norm(r2)^2;
% OMdot = -2*dot(r2,v2)*OM/norm(r2)^2;
% Vrel  = v1-v2-cross(OM,Rrel);
% arel  = a1-a2-cross(OMdot,Rrel)-cross(OM,cross(OM,Rrel))-2*cross(OM,Vrel)

%STATE DERIVATIVE OUTPUT
%Spacecraft Velocity and Acceleration [x;y;z]km/s;[x;y;z]km/s^2
dS1 = [v1;a1];

%Asteroid Velocity and Acceleration   [x;y;z]km/s;[x;y;z]km/s^2
dS2 = [v2;a2];
```

The Absolute to Relative Coordinate Transformation Block

```matlab
function [Srelout,Rrel]    = Abs2Rel(Sout)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.
```

```matlab
r1=Sout(1:3,1);      %Absolute Position of the spacecraft    km
v1=Sout(4:6,1);      %Absolute Velocity of the spacecraft    km
r2=Sout(7:9,1);      %Absolute Position of the asteroid      km/s
v2=Sout(10:12,1);    %Absolute Velocity of the asteroid      km/s

Rrel=r1-r2;          %Relative Position of the spacecraft    km

OM=cross(r2,v2)/norm(r2)^2;     %Rotation of the relative frame s^-1
Vrel=v1-v2-cross(OM,Rrel);      %Relative Velocity of the spacecraft km/s

%Assemble the rotation matrix from the Position and Velocity Vectors
x_test=r2/norm(r2);
z_test=cross(r2,v2)/norm(cross(r2,v2));
y_test=cross(x_test,-z_test);
%Direction Cosine Matrix
Q=[x_test,y_test,z_test]';

%Rotation of the Absolute Frame into the Relative Frame
Srelout=[Q*Rrel;Q*Vrel];
```

The Relative to Absolute Coordinate Transformation Block

```matlab
function Ts   = Rel2Abs(Sout,Trel)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

r1=Sout(1:3,1);      %Absolute Position of the spacecraft    km
v1=Sout(4:6,1);      %Absolute Velocity of the spacecraft    km/s
r2=Sout(7:9,1);      %Absolute Position of the asteroid      km
v2=Sout(10:12,1);    %Absolute Velocity of the asteroid       km/s

%Assemble the rotation matrix from the Position and Velocity Vectors
x_test=r2/norm(r2);
z_test=cross(r2,v2)/norm(cross(r2,v2));
y_test=cross(x_test,-z_test);
%Direction Cosine Matrix
Q=[x_test,y_test,z_test]';

%Rotation of the Relative Frame into the Absolute Frame
Ts=zeros(3,1);
Ts=Q'*Trel;
```

Interface Block

```matlab
function [R,V]    = Int(SC_Position,Srelin)

Rdes=SC_Position(1:3,1);    %Desired Spacecraft Position    km
Ract=Srelin(1:3,1);         %Actual Spacecraft Position     km

Vdes=SC_Position(4:6,1);    %Desired Spacecraft Velocity    km/s
Vact=Srelin(4:6,1);         %Actual Spacecraft Velocity     km/s
R=Rdes-Ract;                %Error in the Position          km
V=Vdes-Vact;                %Error in the Velocity          km/s
```

Acknowledgements

References

[1]Scheeres, D., J., Williams, B., G., and Miller, J., K., *Evaluation of the Dynamic Environment of an Asteroid: Applications to 433 Eros*, Journal of Guidance, Control, and Dynamics, Vol. 23, No. 3, 2000.

[2]Broschart, S., B., and Scheeres, D., J., *Control of Hovering Spacecraft Near Small Bodies: Applications to 25143 Itokawa*, Journal of Guidance, Control, and Dynamics, Vol. 28, No. 2, 2005.

[3]Sawai, S., Scheeres, D., J., and Broschart, S., B., *Control of Hovering Spacecraft Using Altimetry*, Journal of Guidance, Control, and Dynamics, Vol. 25, No. 4, 2002.

[4]Lara, M., and Scheeres, D., J., *Stability Bounds for Three-Dimensional Motion Close to Asteroids*, AAS/AIAA Space Flight Mechanics Meeting, AAS Publications Office, San Diego, CA, 2002.

[5]Abercromby, K., J., "Orbital Perturbations", AERO 557 Course Notes, Lectures 4-6, California Polytechnic State University, San Luis Obispo, CA, 2011.

[6]Mehiel, E., "Phase Plane Analysis", AERO 560 Course Notes, Lecture 10, California Polytechnic State University, San Luis Obispo, CA, 2005.

[7]Broschart, S., B., and Scheeres, D., J., *Lyapunov Stability of Hovering Spacecraft in Time-Invariant Systems*, American Astronautical Society, AAS 05-381.