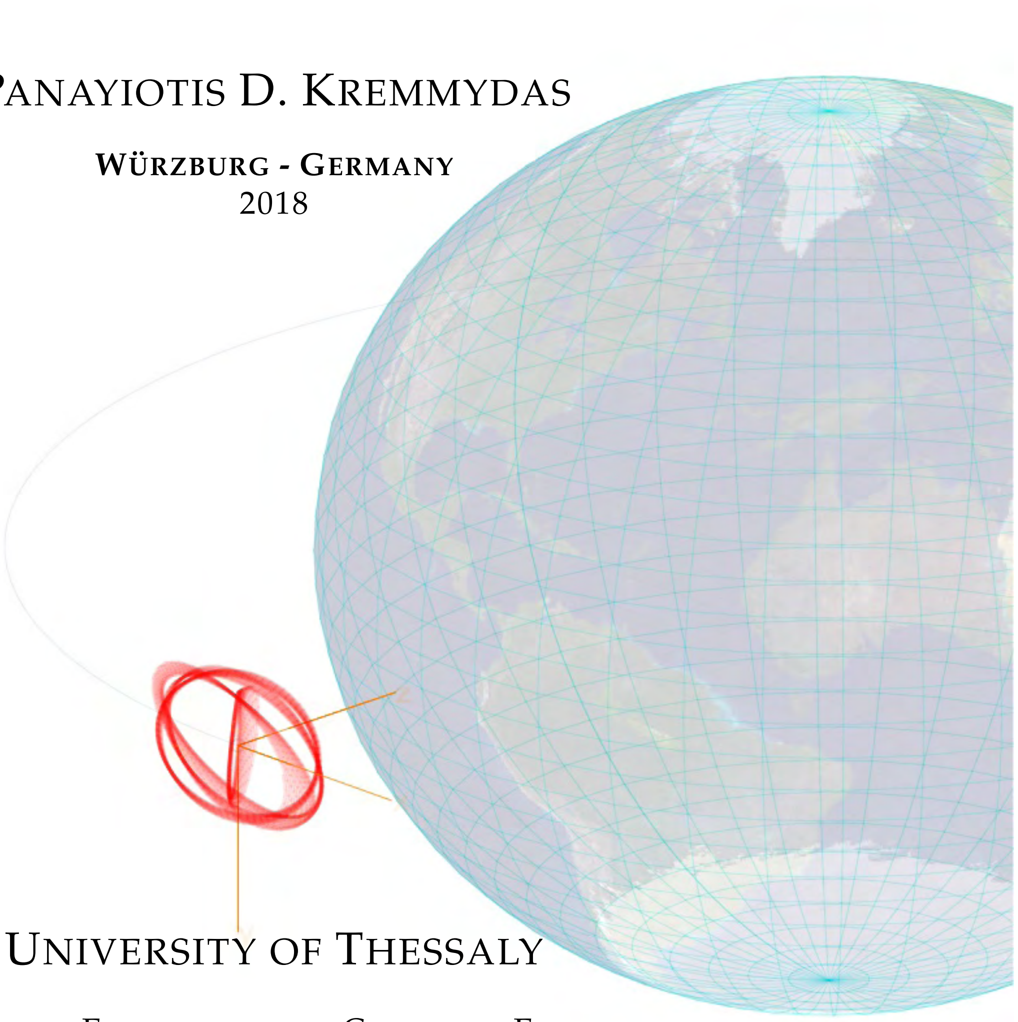
Model Predictive Control for Continuous Low Thrust Satellite Formation Flying

Panayiotis D. Kremmydas

Würzburg - Germany
2018



University of Thessaly

Department of Electrical and Computer Engineering

Zentrum für Telematik e.V.

Aerospace Department

Model Predictive Control for Continuous Low Thrust Satellite Formation Flying

KEYWORDS:MODEL PREDICTIVE CONTROL; SATELLITE FORMATION FLYING; FORMATION CONTROL; CONTINUOUS LOW THRUST; SPACECRAFT AUTONOMY

PANAYIOTIS D. KREMMYDAS

Master's Thesis submitted in fulfillment of the requirements of the University of Thessaly's degree in Electrical and Computer engineering.

Examination date: February 15, 2018.

Περίληψη: Η πτυχιακή αυτή παρουσιάζει μία μεθοδολογία καθοδήγησης και ελέγχου κατανεμημένων συστημάτων δορυφόρων που πετούν σε αποστάσεις κοντινές στην Γη με σκοπό την προσέγγιση διάφορων τρισδιάστατων σχηματισμών εν τροχιά κάτω από τεχνικούς περιορισμούς υπαρχόντων δορυφορικών συστημάτων.

Abstract: *This thesis presents a low thrust guidance and control strategy for distributed spacecraft systems flying in proximity in low Earth orbit with the purpose of converging to arbitrary three dimensional force free formation topologies taking account all possible constraints and system limitations.*

Contents

Introduction

Satellite Formation Flying is promising goal on pursuit. Space science and technology is undergoing various attempts to achieve distributed mission architectures reducing down the costs, development time and expanding possibilities for further mission concepts. Contributing to the improvement of telecommunications, earth and deep space observation missions, the discretization of satellite systems flying in proximity using three dimensional force free formations has yet to prove itself valuable in real world applications.

There have been various successful missions demonstrating accurate stationkeeping like the well known magnetospheric multi-scale (MMS) mission from NASA and other similar formation flying missions in the force induced and force free categories.

One that stands out for demonstrating autonomous maintenance and acquisition of a force free three dimensional formation is the Can-x 4& 5 mission from Deep Space Industries which was the first research mission to form a projected circular orbit formation, specifically for two satellites, at a 50m and 100m ellipse radius.

In the UWE-4 mission, subject of this paper, it is pursued to develop efficient guidance and control to various dynamically evolving topologies as such. More specifically, three dimensional force free formations that are defined from various geometric configurations and consist of four identical spacecraft equipped with 5.33 μN low thrust propulsion and a mass of approximately 1.3 kg.

Although the control task at hand for missions with constant targets is relatively straightforward, demonstrating low-thrust autonomous control on arbitrary satellite states to converge them in such dynamic topologies like the ones planned for the UWE-4 mission, involves a variety of complex challenges.

0.1 Applications and Benefits

There are two important scenarios that fit and increase efficiency in discrete space mission objectives from using a controller able to handle time varying optimal propagated reference states. The first case is when the objectives involve state acquisition for an instantaneous frame, part of an orbit and the second case is when an orbiting formation topology itself is required and a geometrically defined spacecraft configuration needs to perform a task throughout an orbit.

For the first case its possible to achieve with great flexibility location pinpointing with a lower fuel usage for a given distributed satellite system at a given instant of an orbit while keeping the system in a low drift state. Such benefit stems from making station-keeping and altitude convergence fuel usage redundant.

The second case is when three dimensional formation topology is necessary to be acquired as a goal by itself and the convergence to that specified propagated state that defines a given formation and satisfies given mission criteria for the duration of the periodic motion is necessary.

The following forms of target acquisition, including the above, can be applied given various degrees of freedom for the relevant arbitrary target state velocities and matching LVLH dynamic or static topologies formed through them[1].

[1]Assuming no perturbations

Chapter 1

Theoretical background

1.1 Coordinate Systems

In Orbital Mechanics and specifically in Space flight many problems arise concerning the rates with which bodies move in relation to each other and their surroundings. For that purpose there are several different Coordinate Systems describing in the most intuitive manner their given subject of analysis. The term Coordinate System itself adheres to a system that describes a precise location for a given set of numeric values. A Coordinate System is composed by a predefined Reference Frame[1], Coordinate Variable Type and Coordinate Variable Orientation to extract an exact location in space.

Reference Frames

It is well know that a GPS unit needs a signal from at least 4 satellites to pinpoint an exact location in space. A single satellite can give a point within the surface of a sphere as the possible location of the unit, two satellites limit that surface to the line of a circle, three satellites limit that line to

[1]Quite confusingly the terms "Coordinate System"[SCSP08], "Coordinate Frame" [NM13] and "Coordinate Reference Frame"[BH08] share the same meaning throughout the related literature. For the purpose of this thesis and according to the established ISS Programm's consistent terminology, the word "Frame" will only be used to express the Reference Frame definition itself included in the definition of a Coordinate System.

only two possible exact points in space and a fourth satellite if necessary can aid in finding the correct point between them. In a similar fashion, a reference frame needs an origin point and three vectors or equivalently four points in space to define a three axis system and all possible three dimensional locations[2]. A RF is one of the three components that define a Coordinate System. Any motion or the relative properties of a system has to be set in relation to a set of specific points,vectors or axes.

To define a RF around an object for a Space Mission those reference points or reference vectors or directly reference axis are usually formed from: the centre of mass, a point on a surface or centre of volume of interest, axis of rotation, vector of instantaneous velocity direction, vector between two centres of mass, vector from the perceived origin point and perpendicular to the plane defined by the other axes, Vernal equinox sub-solar point of a specific year[3]. A RF can belong to one of the following two categories for a given system:

- **Fixed Reference Frame or Global Frame**: A RF formed by reference points conceived as static to the system's analysis. For example the Sun's or Earth's centre of mass in a heliocentric or geocentric analysis or the Vernal equinox.
- **Moving Reference Frame or Local Frame**: A RF formed by reference points conceived as moving to the system's analysis. For example the centre of mass of a chief satellite in a formation of satellites orbiting a large planetary body.

The reason of this distinction is the interest in the relative motion dynamics themselves in a given set of local objects (e.g. satellite formation) while at the same time uniformly being under major influence by another object (e.g. Earth). A fixed point of reference with respect to a LEO satellite

[2]When it is only the relative distance and velocity to the centre of an object that is useful to an analysis it can be simply inferred to "according to the RF" without any further definition of reference points to complete the frame (e.g."From the Sun's reference frame, the Earth orbits the Sun with an average velocity of 30km/s and distance of 150.000km")

[3]Every year there are two specific times the sun has a sub-solar point (90° angle with the surface of the Earth) on the line of the equator. The precise position of a sub-solar point on the time of Autumn (Vernal) Equinox with the information of the year that this event took place are used by Space Mission Designers as a reference point from which an extension to the centre of the Earth or the Sun (linear extension through the Earth's centre of mass radially towards the surface of the sun) forms one of the reference axes.

formation for simplification purposes is the Earth's CoM. A moving point of reference in proximity of which a satellite formation can be observed and analysed is often defined to be a distinct satellite's CoM, a virtual orbiting point or an algorithmically defined local virtual location[4].

As it is already clearly derived, any motion can be directly expressed according to a Fixed or Global RF of a given system without defining a moving or local reference frame. Any further generalizations of the RF with respect to the Sun, the Galaxy, the Galaxy Cluster or really distant stars which have minuscule effect over the scale of the studied motion can be approximated accurately given the known motion of the celestial bodies.

The preferred method of analysis is setting the reference frame with respect to the object's surroundings locally and simplifying any external effects as certain corrections to the laws of motion. As an example, the non-inertial orbit of a satellite affected by Earth's gravity can be simplified locally as inertial applying the Centrifugal pseudo-force (directed away from the axis of rotation) or in another example in the case of an aircraft moving at much inferior speeds some simple but practical modifications of the gravitational coefficient can help neglect the small effect of the sun's rotation around its axis. In general, when trying to apply the laws of motion in accelerated frames of reference fictitious forces are perceived within tham known as "Inertial Forces" analogous to properties that have to do with the actual body motion in respect to their corresponding fixed coordinate system. In many occasions, working on the basis of a balanced system as if moving at a constant speed greatly simplifies the process of analysis.

Coordinate Variable Types and Coordinate Variable Orientation

There are many types of coordinate variable types used in different contexts (e.g. cylindrical, skew, canonical, trilinear), the ones most commonly used in formation control are the following:

[4]For example, if the sun of a planetary system in addition was having an effect on the same formation with significant variation over the formation's orbit, then the planetary body would be a moving point as well in that specific system analysis, assigning the sun as the fixed point.

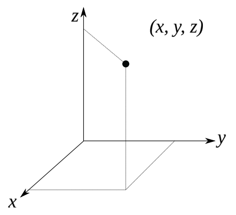


Figure 1.1: Cartesian Coordinate Variables

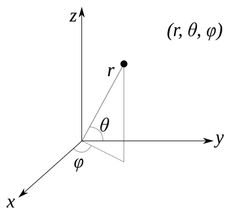


Figure 1.2: Spherical Coordinate Variables

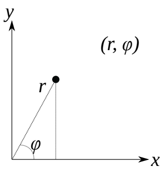


Figure 1.3: Polar Coordinate Variables

Besides defining the RF and variable type of a Coordinate System, the orientation of the measured variables with respect to the RF has also to be defined. Angle θ for example could be measured from 0° to 180° from the north to the south pole, 0° to 180° from the south to the north pole, 90° to -90° degrees with 0° corresponding to coordinates on the equator plane or even defined by an entirely new set of axes derived from the RF of the given system. In the figures below (see Fig. 4), two examples are illustrated having common RF (x,y,z axes) defined in relation to Earth and common Spherical Coordinate Variables (r,θ,ϕ).

Common Coordinate Systems in Space Flight

The selection of Coordinate Systems bellow reflects some of the most commonly used/required for satellite formation flying.

- J2000 Coordinate System [5]

  Often referred to as ECI[6] (Earth-Centred inertial or fixed) Coordinate

[5]Information concerning the exact definition of the mean Equinox time calculation or the mean Equator plane of a referenced epoch and many other properties that precisely define a pre agreed RF such as the J2000 are non-disclosed.

[6]Normally ECI should be the RF category that the J2000 Coordinate System belongs

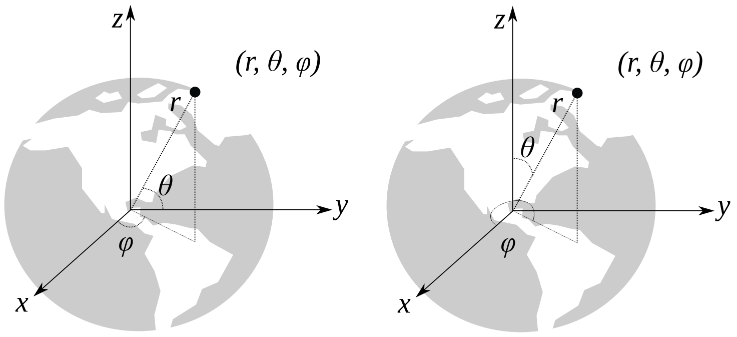


Figure 1.4: Defining the type of variables and a RF is not enough: In the left example θ is measured from the equatorial plane, ϕ incremented in the direction of the Earth's rotation. In the other example: θ measured from the north pole, ϕ incremented in the opposite direction of the Earth's rotation

System. Two axes defined by the sub-solar point precisely on the time of the Vernal Equinox of the year 2000 in the Gregorian Calendar, being a vector directed from the centre of the Earth to the centre Sun, inwards to the solar system through that specific point on the equator. In a spacecraft formation mission its typically used in the launch phase to meet the orbiting parameters of the mission.

Coordinate Variable Type:

Cartesian Coordinate Variables

Reference Frame, Coordinate Variable Orientation: O: Earth's centre of mass

X_{J2000} axis: Formed from the origin to the vernal equinox sub-solar point of year 2000, positive outwards that direction

Y_{J2000} axis: Vector formed from the origin and perpendicular to the plane formed by the X and Z axis, positive in the direction of the **Z** x **X** vector (right handed rule.)

Z_{J2000} axis: Lies on the rotational axis of the year 2000, positive to the north pole.

- TOD (True of Date) Coordinate System

  Defined the same way as the J2000 Coordinate system with the only difference being the year matching the three shifting reference vectors. The changes throughout the years include the slight tilt of the

in, other systems mentioned here are also Earth-Centred Inertial.

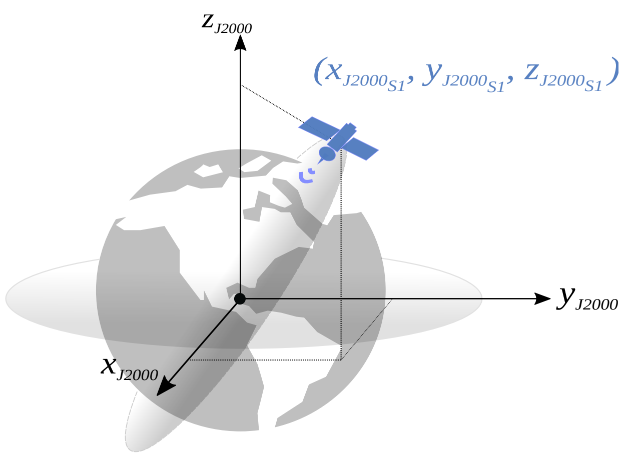


Figure 1.5: J2000 Coordinate System

Earth's rotation axis and the change of the vernal equinox' time, equator plane and corresponding sub-solar spot. It is not as practical as using the J2000 because of the frequency the coordinates have to be transformed each year to correspond to the updated x,y,z reference axes.

- Greenwich TOD Coordinate System.

  Often referred to as Earth-centre Earth-fixed Rotating Coordinate System is the same as the TOD Coordinate System with the only difference being the x axis formed from the centre of the Earth to the point of the prime meridian crossing the True of Date equator instead of a sub-solar point.

- Peri-focal Coordinate System

  Similar to the previous but having its axes defined by the orbit of the spacecraft under analysis. Axis X is set from the centre of mass of Earth to the instantaneous periapsis point of the corresponding orbital body, **z** perpendicular to the instantaneous orbit plane positive towards the north pole and Y positive in the direction of the $\mathbf{z} \times \mathbf{x}$ vector (right handed rule)

- J2000 Polar Coordinate System

  It is the Spherical variation of the J2000 coordinate system, having the same RF as the J2000 and different Coordinate Variable Type and corresponding Orientation.

  Coordinate Variable Type:

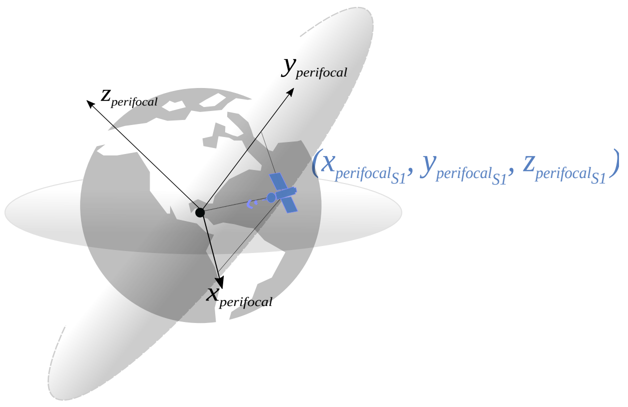

$y_{perifocal}$
$z_{perifocal}$
$(x_{perifocal_{S1}}, y_{perifocal_{S1}}, z_{perifocal_{S1}})$
$x_{perifocal}$

Figure 1.6: Peri-focal Coordinate System

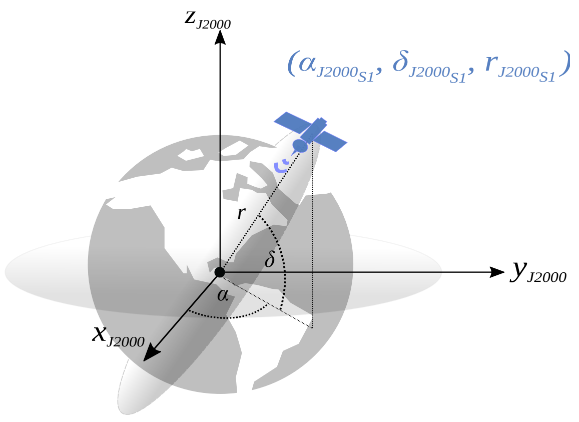

z_{J2000}
$(\alpha_{J2000_{S1}}, \delta_{J2000_{S1}}, r_{J2000_{S1}})$
r
δ
α
y_{J2000}
x_{J2000}

Figure 1.7: J2000 Polar Coordinate System

Spherical Coordinate Variables[7] with right ascension "α" and declination "δ".

Coordinate Variable Orientation:

α_{J2000} Right Ascension: Angle formed from the projection of the radius vector on the X,Y axes plane to the X axis, positive towards the direction of rotation of the Earth.

δ_{J2000} Declination: Angle formed from the X,Y axes plane (mean equator plane) to the radius vector, positive towards the north pole.

r_{J2000} Radius: Distance to the Centre of the Earth.

[7]Although its official declaration[SCSP08] would normally describe "Polar" coordinate variables, that is not the case.

- Local Vertical Local Horizontal (LVLH) Coordinate System[8]

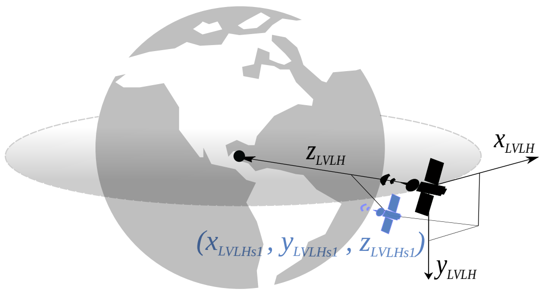


Figure 1.8: Local Vertical Local Horizontal

Often referred to as LO (Local Orbital) Coordinate System, it is the most commonly used moving frame Coordinate System for formation control and will be the main focus on the following chapters.

Coordinate Variable Type:

Cartesian Coordinate Variables

Reference Frame, Coordinate Variable Orientation:

O: Orbiting spacecraft's centre of mass

X_{LO} axis:(Often named as V-bar[Feh03]) Lies on the instantaneous velocity vector, positive in the direction of motion.

Y_{LO} axis:(Often named as H-bar[Feh03]) Perpendicular to the instantaneous orbit plane and directed opposite to the orbit momentum (right handed rule **Z** x **X**)

Z_{LO} axis:(Often named as R-bar[Feh03]) Directed from the origin to the centre of mass of the fixed point of reference

1.1.1 Keplerian two body equations

Adhering to a simplistic model of spherical bodies, Newtonian only gravitational forces and the smaller body having zero or insignificant effect on the larger, the following derivations are extracted using the Geocentric Inertial Coordinate System:

[8]The reader should always be careful of the direction of the three axis in any given reference. There is a Russian and an American equivalent version of the LVLH that have different orientations for the three axis. For the purpose of this thesis, the ESA's and ISS program's standard definition of LVLH described in the current page will be used for the following chapters.

- Orbital period on elliptic and circular orbits.

$$T = 2\pi\sqrt{\frac{a^3}{\mu}}$$

- Total orbital energy

$$\epsilon = -\frac{\mu}{2a}$$

- Angular Velocity.

$$\dot{f} = \sqrt{-\frac{\mu}{a^3(1-e^2)^3}}(1+ecosf)^2$$

- Orbital Velocity.

$$V = \sqrt{\frac{2\mu}{r} - \frac{\mu}{a}}$$

or for circular orbits:

$$V = \sqrt{\frac{\mu}{r}}$$

e: eccentricity:

$$e = \frac{semi - minor \quad axis}{semi - major \quad axis}$$

a: semi-major axis. whereas: (a=r for circular orbits):

$$a = \frac{periapsis \quad distance + apoapsis \quad distance}{2}$$

r: distance from the center of mass or orbit radius in case of circular

The corresponding Velocity Vector in the Perifocal coordinate system takes the following form:

$$[\dot{r}] = \sqrt{\frac{\mu}{a(1-e^2)}}\begin{bmatrix} -sinf \\ e + cosf \\ 0 \end{bmatrix}$$

1.1.2 Hill Equations

The equations of relative motion for circular obits[Hil78] given in the Local Vertical Local Horizontal Coordinate System. The equations are derived assuming a perfectly spherical mass and no perturbations.

$$\begin{aligned}\ddot{x} - \frac{1}{m_c}F_x &= 2\omega\dot{z} \\ \ddot{y} - \frac{1}{m_c}F_y &= \qquad\quad - \omega^2 y \\ \ddot{z} - \frac{1}{m_c}F_z &= -2\omega\dot{x} + 3\omega^2 z\end{aligned}$$

Where ω is the angular frequency of the chief satellite which is assumed to be roughly equivalent to the deputy satellites:

$$\omega = \omega_{chief} = \sqrt{\frac{\mu}{r^3}}$$

1.1.3 Clohessy - Wiltshire Equations

The relative motion of a satellite in the LVLH Coordinate System, given the initial state vector
[x , y , z , $\dot{x}$, $\dot{y}$, $\dot{z}$] and a time invariant propulsion thrust $\vec{\gamma}_(x, y, z)$:

$$x(t) = (\frac{4\dot{x}_0}{\omega} - 6z_0)sin(\omega t) - \frac{2\dot{z}_0}{\omega}cos(\omega t) + (6\omega z_0 - 3\dot{x}_0)t + (x_0 + \frac{2\dot{z}_0}{\omega}) \quad + ...$$
$$\gamma_z \frac{2}{\omega^2}(\omega t - sin(\omega t)) + \gamma_x(\frac{4}{\omega^2}(1 - cos(\omega t)) - \frac{3}{2}t^2)$$

$$y(t) = y_0 cos(\omega t) + \frac{\dot{y}_0}{\omega}sin(\omega t) \quad + \quad \gamma_y \ \frac{1}{\omega^2}(1 - cos(\omega t))$$

$$z(t) = (\frac{2\dot{x}_0}{\omega} - 3z_0)cos(\omega t) + \dot{z}_0\frac{1}{\omega}sin(\omega t) + (4z_0 - \frac{2\dot{x}_0}{\omega}) \quad + ...$$
$$\gamma_x \ \frac{2}{\omega^2}(sin(\omega t) - \omega t) + \gamma_z \ \frac{1}{\omega^2}(1 - cos(\omega t))$$

Closed form solution to the constant input forces in the equations of relative motion[CW60] have been derived from the Hill equations by Clohessy and Wiltshire.

The solutions above are valid only for circular orbits and are linearised[9], meaning that they are good only for small satellite distances as the curvature of the orbit is assumed to have insignificant effect. Further attempts have been successfull in generallizing the equation above to a non-linear solution taking account curvature and generalizing to elliptic motion as well.

For the purpose of this thesis the derived precision of the Clohessy - Wiltshire Equations is sufficient for the relevancy of the results to satellite formations ranging from 0.1-2km distance[10]. As seen in the terms above there is no starting time intruding in the equations (e.g. in relation to a phase of an elliptic orbit) as a result of the second order terms being time-invariant the system belongs to the LTI category having a multitude of analysis and solution tools

An numerically easy way to obtain a solution of the above equation given the differential equations of motion is working in their state space form and solving the system through a Laplace transformation.

[9]assuming that gravitational acceleration is linear and constant across the Cartesian x axis

[10]Existing practical applications in ISS's randevouz and docking missions.

To avoid confusion with the cartesian coordinate variable common notations the state of the system will be denoted with $'x_{state}'$, the input with $'u_{input}'$ and the output with $'y_{output}'$.

General state space form:

$$\dot{x}_{state}(t) = A_c x_{state}(t) + B_c u_{input}(t)$$
$$y_{output}(t) = C_c x_{state}(t) + D_c u_{input}(t)$$

Hill equations in state space form:

$$\dot{x}_{state} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\omega^2 & -2\omega & 0 & 0 \end{bmatrix} * \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \gamma_x(t) \\ \gamma_y(t) \\ \gamma_z(t) \end{bmatrix}.$$

$$y_{output}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \gamma_x(t) \\ \gamma_y(t) \\ \gamma_z(t) \end{bmatrix}.$$

Where the output y is the current state of the spacecraft at a given time ($C_c = I$) and is not linked to the input thrust vector in any way ($D_c = 0$).

Taking the laplace transform of the state space representation above

$$sX(s) - X(0) = A_c X(s) + B_c U(s)$$

$$(sI - A_c)X(s) = X(0) + B_c U(s))$$

$$X(s) = (sI - A_c)^{-1}X(0) + ((sI - A_c)^{-1}B_cU(s))$$

And converting them back to the time domain

$$x_{state}(t) = \mathcal{L}^{-1}\{(sI - A_c)^{-1}\}x_{state}(0) + \mathcal{L}^{-1}\{(sI - A_c)^{-1}B_cu_{input}(t)\}$$

Considering a constant input u(t)=u(0) the equation becomes

$$x_{state}(t) = e^{At}x(0) + e^{A_ct}\int_0^t e^{-A_c\tau}B_cu_{input}(\tau)d\tau$$

$$x_{state}(t) = e^{A_ct}x_{state}(0) + (e^{A_ct}\int_0^t e^{-A_c\tau}B_cd\tau)u_{input}(0)$$

And the solution takes the form

$$x_{state}(t) = \Phi(t)x_{state}(0) + H(t)u_{input}(0)$$

Where

$$\begin{aligned}\Phi(t) = \mathcal{L}^{-1}\{(sI - A_c)^{-1}\} &= e^{A_ct}\\ H(t) = \mathcal{L}^{-1}\{(sI - A_c)^{-1}B_c\} &= e^{A_ct}\int_0^t e^{-A_c\tau}B_cd\tau\end{aligned}$$

For a constant input of magnitude u(0) and an initial position x(0) the above function after time Δt becomes

$$x_{state}(\Delta t) = \Phi(\Delta t)x_{state}(0) + H(\Delta t)u_{input}(0)$$

Extending to discrete time for a sampling interval of Δt

$$x_{state}(k+1) = \Phi(\Delta t)x_{state}(k) + H(\Delta t)u_{input}(k)$$

And written in their discrete state space form for the specific system dynamics $A_d = \Phi(\Delta t), B_d = H(\Delta t), C_d = C_c$ and $D_d = D_c$:

$$\begin{aligned} x_{state}(k+1) &= A_d x_{state}(k) + B_d u_{input}(k) \\ y_{output}(k+1) &= C_d x_{state}(k) + D_d u_{input}(k) \end{aligned}$$

For a typical angular frequency of ω=0.0011 rad/second and a sampling interval of 300seconds the equations of orbital motion in their discrete form for both the above Laplace derived solution and the solution by Clohessy Wiltshire become:

$$\begin{matrix} x(k+1) \\ y(k+1) \\ z(k+1) \\ \dot{x}(k+1) \\ \dot{y}(k+1) \\ \dot{z}(k+1) \end{matrix} = \begin{bmatrix} 1.00 & 0.00 & 0.03 & 279.81 & 0.00 & 94.76 \\ 0.00 & 0.95 & 0.00 & 0.00 & 294.95 & 0.00 \\ 0.00 & 0.00 & 1.15 & -94.76 & 0.00 & 294.95 \\ 0.00 & 0.00 & 0.00 & 0.80 & 0.00 & 0.63 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.95 & 0.00 \\ 0.00 & 0.00 & 0.00 & -0.63 & 0.00 & 0.95 \end{bmatrix} * \begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \dot{x}(k) \\ \dot{y}(k) \\ \dot{z}(k) \end{bmatrix} +$$

$$\begin{bmatrix} 43483.09 & 0.00 & 9507.94 \\ 0.00 & 44620.77 & 0.00 \\ -9507.94 & 0.00 & 44620.77 \\ 279.81 & 0.00 & 94.76 \\ 0.00 & 294.95 & 0.00 \\ -94.76 & 0.00 & 294.95 \end{bmatrix} * \begin{bmatrix} \gamma_x(k) \\ \gamma_y(k) \\ \gamma_z(k) \end{bmatrix}$$

$$y_{output}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \dot{x}(k) \\ \dot{y}(k) \\ \dot{z}(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \gamma_x \\ \gamma_y \\ \gamma_z \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \dot{x}(k) \\ \dot{y}(k) \\ \dot{z}(k) \end{bmatrix}.$$

1.1.4 Relative Dynamic Model Comparison

Clohessy-Wiltshire Model In the current chapter the CW model is developed and discussed as the main model for relative orbit control. The existing assumption on this model that gravity is uniform may work well for the current mission implementation given a 1-2% error tolerance but for missions that involve greater relative distances or more precision, different models have to be evaluated for their effectiveness.

$$
\begin{aligned}
\ddot{x} - \frac{1}{m_c}F_x &= 2\omega\dot{z} \\
\ddot{y} - \frac{1}{m_c}F_y &= -\omega^2 y \\
\ddot{z} - \frac{1}{m_c}F_z &= -2\omega\dot{x} + 3\omega^2 z
\end{aligned}
$$

Tschauner-Hempel Model Even slight eccentricity induces great divergence in relative motion and the Tschauner-Hempel model without adding to much complexity (an a) to the already established Clohessy-Wiltshire can accomodate such motion in the following extended form:

$$
\begin{aligned}
\ddot{x} - \frac{1}{m_c}F_x &= \frac{-\mu}{r^3}x + 2\omega\dot{z} - 2\omega\dot{x} \\
\ddot{y} - \frac{1}{m_c}F_y &= -\frac{\mu}{r^3}y \\
\ddot{z} - \frac{1}{m_c}F_z &= \frac{2\mu}{r^3}z - 2\omega\dot{x} + \omega^2 z \\
\dot{\omega} &= \frac{-2n^2 esin(v + evcosv)^3}{(1-e^2)^3}
\end{aligned}
$$

This simple extension greatly increases the resources necessary for optimal path computation. An MPC implementation of the above model would require far greater hardware capabilities for the introduced nonlinearities concerning the time variant true anomaly v.

Schweighart-Sedwick Model The following equations of relative motion are again given for circular orbits the same reference frame and orientation as the above with a difference in the coordinate variable type. The coordinate variables used are curvilinear and more specifically the x and y vectors of the Schweighart-Sedwick coordinate system curve along a sphere of radius equal to the one used by the assumed reference orbit.

$$
\begin{aligned}
\ddot{x} - \frac{1}{m_c}F_x &= \ 2\omega\dot{z} \\
\ddot{y} - \frac{1}{m_c}F_y &= 2lq \ \ cost(qt + \phi) - q^2 y \\
\ddot{z} - \frac{1}{m_c}F_z &= -2\omega\dot{x} + (5\omega^2 - 2n^2)z
\end{aligned}
$$

Such model eliminates the non-linearity errors induced by the above equation but introduces a lot of new terms. The new terms introduced l, q, c, n, ϕ correspond to the following quantities and orbital characteristics:

l : linearized rate of change for the amplitude cross-track separation

q : linearized argument of the cross-track separation ???? angle of cross track separation.

n : mean orbital motion in deg/day

c : $c = \frac{\omega}{n}$ or $c = \sqrt{1+s}$ where s is the most complicated thing in the world ϕ : initial phase angle of the cross-track motion.

Including the effects of earths characteristic geopotential J2, the model has to be corrected model with extra terms defining a constant J2 drift over an orbit [SS02].

Chapter 2

Control Methods and Algorithms

2.1 Analytical solution with Optimal Control

Minimizing the state of error given from the navigational systems of a satellite to its position in LVLH orbit according to its predefined motion within a formation structure is a task easily performed by a typical trajectory control algorithm tuned to the physical constraints of the system. A problem of greater importance that remains to be solved in practice is the acquisition of a new formation structure for a given number of satellites in the most efficient possible way according to the same constraints.

For the following analysis we are assuming that in a typical spacecraft system the value on which there is absolute control is the propulsion itself introduced as force $(\gamma_x, \gamma_y, \gamma_z)$ in the CW equations. As force itself affects directly changes in velocity $(\dot{x}, \dot{y}, \dot{z})$ and indirectly changes in position (x, y, z) a typical state vector for a spacecraft that can fully describes its guidance properties is composed by both the position and velocity vectors $(x, \dot{x}, y, \dot{y}, z, \dot{z})$ [1]

Tackling this problem analytically in the current section, we assume that the only constraints existing in the acquisition of a given trajectory are the maximum possible omnidirectional thrust, the time necessary to com-

[1]In the case that control has to be excerted through controlling a higher order derivative of the position vector (e.g.$\ddot{\vec{p}}$, like in the case of controlling the input fuel flow as a function of time under certain restrictions), the state vector would include higher order variables (e.g. $x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, z, \dot{z}, \ddot{z}$)

plete the maneuver and we simplify that the cost of the manuever we want to minimize is derived directly and proportionally by the given acceleration values through time. We are essentially not taking account the efficiency curve or any special characteristics of the propulsion system used, the instantaneous attitude of the satellite which influences thrust direction and overall efficiency or any given perturbations existing throughout the trajectory in low earth orbit.

Finding an optimal solution for the thrust function F(t) of that maneuver analytically for a single satellite involves the simplification of that function to a polynomial of a predefined order 'n'.

$$\vec{F}(t) = \vec{c_n}t^n + \vec{c_{n-1}}t^{n-1} + ... + \vec{c_1}t + \vec{c_0}$$

Any formation acquisition transfer within a timeframe of execution T has a minimum possible total energy consumption for the acquisition of the final state at the maximum of the allowed timeframe.

Physical Constraints (or State Constraints)

The final conditions that satisfy the new desired state within the new formation structure to be acquired are defined by the final position r_{final} and final velocity v_{final} vectors whereas the initial position $r_{initial}$ and the initial velocity $v_{initial}$ replace the unknown coefficients produced by the following integrals of acceleration α(t) and velocity v(t):

Motion under negligible gravitational forces:

$$\vec{v}_{final} = \int_0^{\tau} \frac{1}{m}\vec{F}(t)dt + \vec{v}_{initial}$$
$$\vec{r}_{final} = \int_0^{\tau} (\int \frac{1}{m}\vec{F}(t)dt + \vec{v}_{initial})dt + \vec{r}_{initial}$$

Which can also be written because of the proportional relationship $\vec{\alpha}(t) = \frac{1}{m}\vec{F}(t)$:

$$\vec{v}_{final} = \int_0^{\tau} \vec{\alpha}(t)dt + \vec{v}_{initial}$$

$$\vec{r}_{final} = \int_0^{\tau} (\int \vec{\alpha}(t)dt + \vec{v}_{initial})dt + \vec{r}_{initial}$$

LVLH orbital motion under gravitational forces:

$$v_{x_{final}} = \dot{x}_{final} = \int_0^{\tau} (\frac{1}{m}F_x(t) + 2\omega\dot{z})dt + v_{x_{initial}}$$

$$r_{x_{final}} = x_{final} = \int_0^{\tau} (\int (\frac{1}{m}F_x(t) + 2\omega\dot{z})dt + v_{x_{initial}})dt + r_{x_{initial}}$$

$$v_{y_{final}} = \dot{y}_{final} = \int_0^{\tau} (\frac{1}{m}F_y(t) - \omega^2 y)dt + v_{y_{initial}}$$

$$r_{y_{final}} = y_{final} = \int_0^{\tau} (\int (\frac{1}{m}F_y(t) - \omega^2 y)dt + v_{y_{initial}})dt + r_{y_{initial}}$$

$$v_{z_{final}} = \dot{z}_{final} = \int_0^{\tau} (\frac{1}{m}F_z(t) - 2\omega\dot{x} + 3\omega^2 z)dt + v_{z_{initial}}$$

$$r_{z_{final}} = z_{final} = \int_0^{\tau} (\int (\frac{1}{m}F_z(t) - 2\omega\dot{x} + 3\omega^2 z)dt + v_{z_{initial}})dt + r_{z_{initial}}$$

The thrust in this case is not proportional to the acceleration in a given axis (e.g. $F_x \neq m\ddot{x}$) but it is contributing to motion as described by the Hill Equations (e.g. $F_x = m(\ddot{x} - 2\omega\dot{z})$) for the corresponding axis to be enforced on. [2]

Whereas having an orbital speed $\omega \to 0$ is essentially reducing down the above equations to the well known gravity free variant.

Control Constraints

The constrain of the thrust function $F(t)$ to the maximum possible values that the propulsion can enforce within the timeframe τ is derived by: [3]

[2] Any possible orbital petrubations can be written as separate thrust functions into the Hill equations and get easily implemented into the state space equations.

[3] Having a specific thrust profile that the propulsion system enforces can be implemented as separate time variant functions $A(t), B(t), C(t)...etc$ that constraint an area within which the thrust function can be formed in

$$|\vec{F}(t)| \leq A \, , \quad 0 \leq t \leq \tau$$

Performance Measure

Finally, the function that needs to be minimized for the maximum possible fuel efficiency within the sets of possible thrust function coefficients derived from the above constraints is the following: [4]

$$Minimum \quad of: \quad \int_0^{\tau} |\vec{F}(t)| dt, \quad 0 \leq t \leq \tau$$

2.2 Modern Conrol Methods

The state space form of the system as described by the CW equations is given in the following form:

$$x_{state}(k+1) = Ax_{state}(k) + Bu_{input}(k)$$
$$y_{output}(k+1) = Cx_{state}(k)$$

$$y_{output}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \dot{x}(k) \\ \dot{y}(k) \\ \dot{z}(k) \end{bmatrix}$$

....

[4]For an approach that includes derived fuel costs from a magnitude defined fuel efficiency function of thrust (e.g $cost(\vec{F}(t)) = |a\vec{F}^2(t) + b\vec{F}(t)| + c)$, the appropriate function has to be used inside the integral (e.g. $\int_0^{\tau} cost(\vec{F}(t))dt$) and for reduced computational complexity , it is wise to divide the force function to its negative and positive equivalent $(F(t) = F_{pos}(t) + F_{neg}(t))$ dealing with a single integral result to be minimized (e.g. $\int_0^{\tau}(F_{pos}(t) - F_{neg}(t)) \quad dt, \quad 0 \leq t \leq \tau$)instead of conditional results over the arbitrary coefficient sets derived from an absolute function

The system is controllable by its definition [5].

Where the cost function to be minimized for 6 state variables and 3 inputs without intermidiate states having effect has the general form:

$$J^*_{0,N} = \begin{bmatrix} x_{target}(N) - x(N) \\ y_{target}(N) - y(N) \\ z_{target}(N) - z(N) \\ \dot{x}_{target}(N) - \dot{x}(N) \\ \dot{y}_{target}(N) - \dot{y}(N) \\ \dot{z}_{target}(N) - \dot{z}(N) \end{bmatrix}^T W \begin{bmatrix} x_{target}(N) - x(N) \\ y_{target}(N) - y(N) \\ z_{target}(N) - z(N) \\ \dot{x}_{target}(N) - \dot{x}(N) \\ \dot{y}_{target}(N) - \dot{y}(N) \\ \dot{z}_{target}(N) - \dot{z}(N) \end{bmatrix} + \sum_{n=0}^{N-1} \left\{ \begin{bmatrix} \gamma_x(k) \\ \gamma_y(k) \\ \gamma_z(k) \end{bmatrix}^T H \begin{bmatrix} \gamma_x(k) \\ \gamma_y(k) \\ \gamma_z(k) \end{bmatrix}^T \right\}$$

Customized weight function placing different importance on every value:

$$W = \begin{bmatrix} W_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & W_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & W_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & W_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & W_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & W_6 \end{bmatrix}, H = \begin{bmatrix} H_1 & 0 & 0 \\ 0 & H_2 & 0 \\ 0 & 0 & H_3 \end{bmatrix}.$$

For the specialized case of minimizing the distance to a specific target state and its associated thrust cost having equal weight on all axes:

$$J^*_{0,N} = R_1 \left\| \begin{matrix} x_{target}(N) - x(N) \\ y_{target}(N) - y(N) \\ z_{target}(N) - z(N) \\ \dot{x}_{target}(N) - \dot{x}(N) \\ \dot{y}_{target}(N) - \dot{y}(N) \\ \dot{z}_{target}(N) - \dot{z}(N) \end{matrix} \right\|^2 + R_2 \sum_{n=0}^{N-1} \left\| \begin{matrix} \gamma_x(k) \\ \gamma_y(k) \\ \gamma_z(k) \end{matrix} \right\|^2.$$

With R1 and R2 defined as constants. R1 and R2 define the relationship of importance between meeting the specific target state requirements and input thrust cost for formation acquisition. The weights can be functions of time and the cost function can be altered within operation meet shifting demands, like in formation control initially the target state to be achieved might have less importance than fuel cost but as time continues until the

[5]Every state can be manipulated by the system's input. The spacecraft's position and velocity can be manipulated by its thrusters.

set acquisition time the importance of meeting the final state might be increased.

The quadratic form on the L1 of the distance from the target and the thrust values is used to negate the root which does not affect minimization and makes it easier to handle mathematically.

———-

2.3 Linear Quadratic Regulator

Optimal Control with a cost function that has terms on a quadratic power.[6]. A real-valued function defined on an n-dimensional interval is called convex, the case of the Linear Quadratic Regulator is useful when the set of the differential equations is linear and the cost derived from the system dynamics is approximated by a quadratic function of the following form:

$$J = x^2(t_{target}) + \int_{t_{initial}}^{t_{target}} (x^2(\tau) + u^2(\tau) + 2x(\tau)u(\tau))d\tau$$

The resulting feedback controller can be derived directly from the following equations:

Result is a matrix that you multiply the current state with to get the optimal input necessary for the system to follow the optimal trajectory

Linear Quadratic Gaussian control

Derived from the method above, this extra methodology deals with systems that are uncertain, disturbed by Gaussian noise and having incomplete state information. In the particular case of spacecraft formations the above problems do not need to be addressed and the Linear regulator problem itself is enough

———

[6]Quadratic power becose it can simplify Quadratic Programming solutions[BBC12].

2.4 Model Predictive Control

Model Predictive Control or Receding Horizon Control is a formulated extension of well established Optimal Control whereas the optimal input is found according to various system constraints and a weight defined cost criterion. The main difference lies in the receding nature of the controller which essentially involves iterative re-planning and re-optimization of the control input for the lowest cost target acquisition throughout the moving horizon window. Any model mismatches, manipulated input and measured output disturbances that cause accumulated target deviation over time are thus taken account in such iterative manner. The recent popularity of MPC in engineering has led to developed theory behind it in many cases being driven by control engineers themselves and project oriented implementations.

Since the reciding horizon is essentially calculating Optimal control for trajectories that need to reach a destination within $T_{horizon} < T_{acquisition}$ the controller is enforcing steeper acceleration to meet the required final state. Although getting closer to the final state in the first portion of the trajectory compared to a full horizon optimal control $T_{horizon} = T_{acquisition}$ is something that can be desired, it will always leads to an increased fuel cost assuming that the propulsion systems have a quadratic "cost to acceleration" function because of the decreased initial acquisition timeframe.

Less frequent updating: When the effects of stochasticity in a model are within tolerance for a given timeframe or, simply put, when the amount of noise that builds up over a certain period is quite small there is less need to update the control input values derived from an optimal solution calculation in every single time interval.

The Optimal Control inputs apart from the first one are typically discarded and for every discrete time-step the Optimal Control Problem has to be solved. Less frequent updating is essentially reducing computations by using more than one of the Control inputs calculated by an iteration of the Optimal Control problem before recalculating for the receding horizon. Less frequent updating can be used to reduce computation usage when the system does not have sadden dangers or obstacles within a specific timeframe that cannot be corrected or avoided by control inputs and at the same time reducing the sampling frequency of the system state inputs and frequency of the control outputs would result in high frequency degradation and potential higher cost control.

It has to be noted that in any case the computational load of the Optimal Control problem has to be complete in a time less than the sampling frequency of the system, meaning that it can only help computationally in cases of shared CPU applications or energy dependant systems [7].Multi-step MPC does not extend allowed execution time for the Optimal Control Algorithm unless longer update intervals are implemented on optimal control that gives a stack of optimal control inputs that will be enforced with a delay.

Explicit MPC. What has been first introduced in relevant literature as Dynamic Programming with constraints and later named Parametric Programming or multi-parametric Programming. The result is a simple use of a QP algorithm to solve off-line all possible discrete states $x(k)$ for a given frame k.

Can deal with every possible discrete control system problem iteratively forming a N+1 dimensional vector map of associated states and input controls that point to their derived next state in the minimum cost trajectory that satisfies the cost function and the given constraints (the remaining cost of that trajectory all the way to the end can also be included as redundant information). The extra dimension denotes the discrete steps of control and for example in a three dimensional motion of a spacecraft through time it can denote time increments and their associated optimal control input for each possible state.

The main issue that occurs implementing dynamic programming is the great magnitude of the required memory necessary for storing all this information for a N+1 dimensional control problem. For a spacecraft for example the required memory to control LVLH space motion with a magnitude of 10km devided in discrete points of 1 meter distance in each dimension and a maximum control timeframe of 1000minutes with a 1 minute control interval is resulting in memory that occupies $10^4 * 10^4 * 10^4 * 10^3 = 10^{15}$ slots [8]. Since memory resources are scarce in satellite missions, the explicit implementation of MPC is not considered.

[7]Alternatively, time delaying methods have to be utilized

[8]Some algorithms and methods that can solve the above memory bound issue with certain trade-offs like state Increment Dynamic Programming and Lagrange Multipliers Polynomial Approximation.

Quadratic optimization problem

Solving the Optimal Control problem to derive a trajectory that satisfies the final state and thrust constraints with a minimum cost involves simplifying the cost function to a quadratic form turning the problem into a linear quadratic regulator problem. The LQR problem then has to be solved iteratively within tight sample rates highlighting the need for an efficient and low throughput Quadratic Programming algorithms that forms the main computation load of LQR.

The solution to an MPC control iteration comes down to solving a QP problem. There are several methods used for solving that problem and although minimizng a convex quadratic function is something that has been researched to be solved since 1955 it has been major subject for ongoing research in resource efficient algorithms that can be implemented in the current low cost microelectronic solutions widely available, achieving energy and resource efficient control for remote and discrete applications. The current most prominent methods for solving the QP of an MPC controller are: Active Set methods (best for small and medium sized problems), Interior Point methods (can deal with large scale problems), Gradient projection methods(based on the gradient projection method and uses a limited memory BFGS matrix to approximate the Hessian of the objective function), Conjugate gradient methods, Fast Dual Gradient Projection methods, Forward-Backwards Newton, Augmented Lagrangian methods (or alternating direction Lagrangian multipliers).

Chapter 3

Formation Acquisition

The definition of a spacecraft formation is given by the following statement proposed by NASA's Godard space Flight Center:

"The tracking or maintenance of a desired relative separation , orientation or position between or among spacecraft."

The studied formations in this particular thesis are specifically oriented towards a periodic time shifting separation of satellites to a referrence orbit which generate a symbolic three dimensional structure.

This virtual structure can take an arbitrary number of forms to accomplish a given task. Specifically for formations flying force free in low earth orbit the following formations derived from the Clohelsy Wilshire equations and specific predefined initial conditions can be performed by an arbitrary number of satellites.

3.1 Structural degrees of freedom and acquisition cost efficiency

We are assuming that the spacecraft originating from arbitrary positions have to acquire new state vectors that result in a new trajectories within a specific timeframe. In the special case of motion in an LVLH orbital frame the new state vectors to be acquired can also result in periodical trajectories with a period *"T"* corresponding to the characteristics of the orbit as in the examples given in the previous chapter.

3.1.1 Infeasible Acquisition

Finally after calculating all possible constellation transfers to the specified formation structure it can also be concluded that a feasible control input for the multiple admissible trajectories in each case cannot be satisfied within the required timeframe. The action necessary to be followed in the case above is the calculation (maybe numberically) of the minimum timeframe that the given formation acquisition can be feasible upon which relevant actions can be decided by mission control.

Setting narrow timeframes for formation acquisition is greatly increasing fuel usage. Assuming that any states between the initial state and the target state of each spacecraft are not affected by any special attribute of the system (e.g. propulsion plumes, collision avoidance) wider timeframes always result in lower fuel usage for a given formation acquisition.

3.1.2 Numerical solution with standard MPC algorithmic approaches

Such simplifications are necessary to easily manage coordinates, defining for example the distance to the chief satellite, on which precise calculation and control has to be exerted to satisfy the given motion criteria. It is paramount therefore that any intricacies derived from the non-inertial motion of the satellites be precisely integrated into a translation matrix (taking such corrections into account), upon which the satellites propulsion system for example acts upon to perform the desired manoeuvres that satisfy the local formation control requirements.

As explained in the Fixed RF definition, the orbit of such a satellite formation in LEO is viewed as non-inertial with respect to Earth only and not any further generalized RF because any effects emanating from the differentiating distance to other celestial bodies are considered minuscule and within margin of error; their effect is many orders of magnitude smaller.

3.2 Types of Acquisitions and corresponding LVLH topologies generated

Simple target acquisition: Each satellite is guided to the defined topology having a complete freedom over the target state velocity for the given time constraints. The spacecrafts can acquire the target states optimally with the minimum possible fuel consumption but the force free propagation of those states lead to over time accumulated drift in the LVLH coordinate system and thus there is no resulting LVLH topology.

Drift-free target acquisition or (Force free dynamic unspecified topology acquisition):[1] The possible state velocities in this category are a subset of the simple target acquisition such that the resulting final state propagation has the effect of canceling its own drift. The spacecraft will be reacquiring the same pattern in that particular defined phase every period of the orbit but in the rest of the orbit they will have a topology which can be overextending and amorphous.

Constrained drift-free target acquisition (or Constrained Force free unspecified topology acquisition) A subset of the previous set of velocities that within further mission criteria will not allow orbit in overextending distances from the center of the LVLH frame. The possible amorphous topology created is confined into a constrained LVLH space.

Force free three-dimensional formation acquisition[2] (or Force free specified topology acquisition): This is the case that there are extra rules that define and give relation or symmetry to some or all of the states of the satellites throughout the orbit itself.If the topology is defined within the same constraints as the above category then it is a subset of the previous possible set of ending state velocities. The satellites now orbit in a topol-

[1]Force free describes a constant state in a formation where theoretically no force is necessary to be used and Drift free describes a maneuver that does not allow substantial drift.

[2]A formation is a dynamic topology evolving with specifications in its geometric orientation and distance through its individual discrete periodic states. It can be a rigid formation, having constant distances between the spacecraft, or a non-rigid formation that includes a more complicated rule-set as seen in the mission examples below.

ogy with one or more specified properties like the ones presented in the paper that are constantly maintained in a force free manner.All force-free formations are dynamically evolving topologies in the LVLH coordinate system.

Force free one-dimensional formation acquisition (or force-free station-keeping) An additional force-free formation that is not three-dimensional is the well known along the orbit formation[3] or trailing formation, which is defined by station-keeping in specific locations in the x-axis of the LVLH coordinate system.

Forced induced formation acquisition (or formation acquisition): With a given unlimited fuel capacity, any kind of topology and any specified properties can be achieved in the LVLH frame. Force free formations are a subset of this category.

Force induced station-keeping formation acquisition. Formation topology defined by having its individual discrete state velocities to be zero, requiring fuel to be utilized continuously to maintain such state.

Based on the above acquisitions and possible fuel limitations the two most beneficial cases belong to the general maneuver category of Constrained Drift-free Target acquisition, the Force free formation acquisition or a combination of the two. The first case is where a particular state with specific criteria needs to be acquired in a given instant of an orbit, the second case is where given criteria have to be fulfilled throughout an orbit and the third case is a combination of the first two that require convergence to a specific instantaneous state but with some kind of additional definition for the orbit propagation of one or more discrete parts of the satellite system. Such thing can be necessary considering that missions often need to serve multiple additional functions. All three of those cases can be tackled by the same controller, subject of this paper, given a proper way to generate the target trajectory to the optimized controller for each individual satellite.

[3]It includes the specific case that all the satellites orbit along the same altitude through time

3.3 Drift-free target acquisition for a mission objective

To elaborate on a possible mission objective that can give a good idea beyond stereoscopic earth observation we assume the analysis of a particular space target that has to be pointed at from a discrete multi-satellite system in a specific configuration. It can be that for maximum signal resolution, some of the satellites have to form a perpendicular surface and be equidistant to the targets location for a given optimal phase of the systems' orbit.

Giving an example of how the controller works for the first application case, the resulting coefficients for a given satellite objective can often be a set of position state variables in an LVLH frame (X,Y,Z) and a phase in the orbit of a satellite constellation.

The first step of a drift free target acquisition involves strategically choosing the right X,Y,Z velocities that constraint the resulting topology to be drift free and require the least amount of fuel. The propagation of that fully defined LVLH state is then feasible for every satellite in the orbit.

The second step involves using the set of those time-varying target states with a potent controller able to handle dynamic reference trajectories given the existing system limitations. The controller needs to accurately track the calculated dynamic reference vector with a valid time shift concerning the initial phase of the orbit and precisely start control on the exact time the configured sampling frequency allows reaching that state with high accuracy.

Existing common methods for doing such a feat would require nullification of the LVLH velocities and station-keeping costs. The resulting benefit of such maneuver in contrast to current approaches is the fuel savings linked to redundant velocity nullification and station-keeping or savings from the reconfiguration necessary for other objectives that could have been benefited with existing non-zero system velocities or simply savings from the necessity to return back every individual satellite to a drift free state.

3.4 Degrees of freedom in a force free formation

For the second application case, where the formation topology itself is the goal, there can be further freedom in the possible set of target reference states propagated in the controller. Any form of freedom can be used as an arbitrary input to enforce optimization on, for the thorough calculation of the most efficient transfer of a constellation of spacecraft. This essentially means that the minimum cost criteria for the sum of the spacecraft acquisitions is possible to be achieved within a wider set of possible solutions if the mission allows it and a potential reduced cost. The degrees of freedom for a force free formation can be the following:

In a given mission the three degrees of freedom can be spacecraft configuration, a shifted topology location in any of the three axes or simply a shifted phase for the total periodic motion of the topology.

Shifted phase for the total constellation if there are strict rules that define the admissible phase difference and dependent states for the whole constellation of spacecraft (e.g. strictly phase correlated Equidistant formation as seen in the paragraphs below) or a combined set of possible shifted phases for different subsets of the discrete satellite system.

Multiple configurations of the spacecraft within the topology to be acquired (e.g. spacecraft A could potentially interchange positions with spacecraft B in the new formation if the initial positions allow a smaller transfer cost) for every set of spacecraft that share the shame role and can interchangeably fulfill mission objectives in different configurations.

Shifted topology location if the given orbit of the whole constellation upon which the LVLH coordinate system is defined is not set on strict constraints and the entire topology can be shifted in one or more axes.

Summing up, the degrees of freedom upon which the set of possible ending sates is formed within formation acquisition time τ for a constellation are: the initial phase of the formation topology to be acquired at time τ, the spacecraft configuration within that new formation topology and finally a possible shift of the formation topology itself within the LVLH frame.

As a final remark, if fuel balancing is a concern inside a constellation, arbitrary weights in the cost function of every spacecraft inversely proportional to its fuel tank sufficiency can be implemented to increase fuel consumption on the spacecraft that have ample fuel and decrease consumption on the ones that have low reserves.Every possible cost estimation for the given degrees of freedom has to be calculated for the optimal solution to be found according the fuel consumption policy (balanced consumption, strict optimal consumption, spacecraft importance balancing etc.)

Chapter 4

Evaluation

4.1 Model Description

The equations of relative motion for circular orbits [Hil78] given in the Local Orbital Coordinate System.

$$\begin{aligned}
\ddot{x} - \frac{1}{m_c}F_x &= 2\omega\dot{z} \\
\ddot{y} - \frac{1}{m_c}F_y &= \qquad - \omega^2 y \\
\ddot{z} - \frac{1}{m_c}F_z &= -2\omega\dot{x} + 3\omega^2 z
\end{aligned}$$

Hill equations in state space form:

$$\dot{x}_{state} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2\omega \\ 0 & -\omega^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\omega^2 & -2\omega & 0 & 0 \end{bmatrix} * \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} + ...$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \gamma_x(t) \\ \gamma_y(t) \\ \gamma_z(t) \end{bmatrix}.$$

Where ω is the average angular frequency of the constellation of spacecraft and for the UWE-4 mission with an altitude of 700km is:

$$\omega = \sqrt{\frac{\mu}{r^3}} = 0.000001127 rad/sec$$

Closed form solution to the constant input forces in the Hill equations of relative motion by Clohessy and Wiltshire[CW60].

$$\begin{aligned}
x(t) &= (\frac{4\dot{x}_0}{\omega} - 6z_0)sin(\omega t) - \frac{2\dot{z}_0}{\omega}cos(\omega t) + (6\omega z_0 - 3\dot{x}_0)t \\
&+ (x_0 + \frac{2\dot{z}_0}{\omega}) \ + \gamma_z \frac{2}{\omega^2}(\omega t - sin(\omega t)) \\
&+ \gamma_x(\frac{4}{\omega^2}(1 - cos(\omega t)) - \frac{3}{2}t^2) \\
y(t) &= y_0 cos(\omega t) + \frac{\dot{y}_0}{\omega} sin(\omega t) \ + \ \gamma_y \ \frac{1}{\omega^2}(1 - cos(\omega t)) \\
z(t) &= (\frac{2\dot{x}_0}{\omega} - 3z_0)cos(\omega t) + \dot{z}_0 \frac{1}{\omega} sin(\omega t) + (4z_0 - \frac{2\dot{x}_0}{\omega}) \\
&+ \gamma_x \ \frac{2}{\omega^2}(sin(\omega t) - \omega t) + \gamma_z \ \frac{1}{\omega^2}(1 - cos(\omega t))
\end{aligned}$$

Where it is clear that the dynamics of the used system given certain initial conditions lead to periodic motion in the LVLH frame.

4.2 Target Force free three dimensional formations

For the UWE-4 mission the initial topologies to be tested are the following:

3 in plane, 1 our of plane

Defined by 3 satellites having a periodic motion in the X,Z plane and 1 satellite having a periodic motion in the Y plane. Initial conditions:

3 in plane satellites:

$$\begin{aligned} x_0 &= arbitrary = \pm 1000m \\ y_0 &= 0m \\ z_0 &= 0m \\ \dot{x}_0 &= 0m/s \\ \dot{y}_0 &= 0m/s \\ \dot{z}_0 &= (x_0)\frac{1}{2}\omega = 0000m/s \end{aligned} \tag{4.1}$$

(the rest of the states calculated with a relevant force free time propagation equal to a multiple of 1/3 and 2/3 of the orbital period)

1 out of plane satellite:

$$\begin{aligned} x_0 &= 0m \\ y_0 &= arbitrary = 600 \\ z_0 &= 0m \\ \dot{x}_0 &= 0m/s \\ \dot{y}_0 &= arbitrary = 0m/s \\ \dot{z}_0 &= 0m/s \end{aligned} \tag{4.2}$$

Equidistant

Defined by all the satellites having a periodic motion that propagates them with constant distances to eachother through time.Initial conditions:

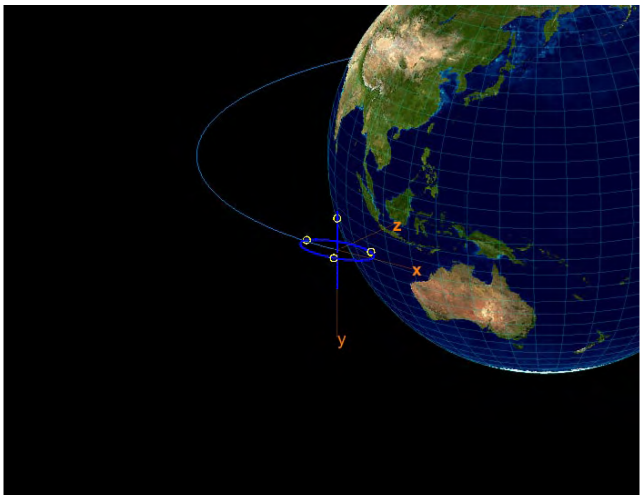


Figure 4.1: 1 in plane-3 out of plane formation

$$\begin{aligned}
x_0 &= arbitrary = \pm 1000m \\
y_0 &= 0m \\
z_0 &= 0m \\
\dot{x}_0 &= 0m/s \\
\dot{y}_0 &= (x_0)\sqrt{\frac{3}{4}}\omega = 0000m/s \\
\dot{z}_0 &= (x_0)\frac{1}{2}\omega = 0000m/s
\end{aligned} \tag{4.3}$$

(the rest of the states calculated with a relevant force free time propagation equal to a multiple of 1/4, 2/4 and 3/4 of the orbital period)

Tetrahedral formation

Defined by two stationary and two dynamic satellites in a way that,through their periodic motion ,a dynamic tetrahedral of constant volume is formed between them throughout the orbit. Initial conditions:

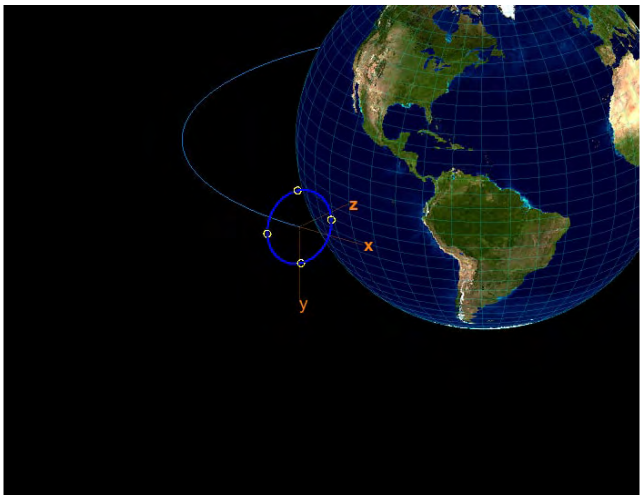


Figure 4.2: Equidistant formation

First Satellite:

$$
\begin{aligned}
x_0 &= \frac{-2}{3}A = \pm 000.90m \\
y_0 &= \sqrt{3}A = 500m \\
z_0 &= \frac{2\sqrt{2}}{3}A = 000m \\
\dot{x}_0 &= \frac{4\sqrt{2}}{3}\omega A = 000m/s \\
\dot{y}_0 &= 0m/s \\
\dot{z}_0 &= \frac{1}{3}\omega A = 000m/s
\end{aligned}
\tag{4.4}
$$

Second Satellite:

$$
\begin{aligned}
x_0 &= 2A = \pm 000.90m \\
y_0 &= \sqrt{3}A = 500m \\
z_0 &= 0m \\
\dot{x}_0 &= 0m/s \\
\dot{y}_0 &= \frac{2\sqrt{2}}{\sqrt{3}}\omega A = 000m/s \\
\dot{z}_0 &= -\omega A = 0000m/s
\end{aligned}
\tag{4.5}
$$

LVLH Stationary satellites:

$$\begin{aligned} x_0 &= arbitrary = \pm 1000m \\ y_0 &= 0m \\ z_0 &= 0m \\ \dot{x}_0 &= 0m/s \\ \dot{y}_0 &= 0m/s \\ \dot{z}_0 &= 0m/s \end{aligned} \tag{4.6}$$

Where $A = \frac{3\sqrt{3}}{40}(semimajor) = 129.90$;

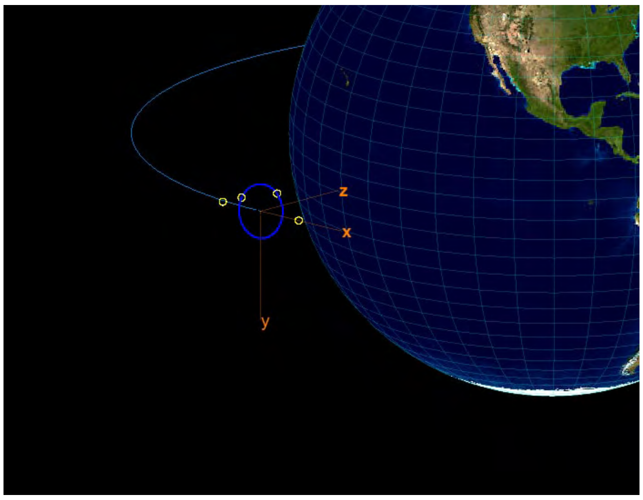


Figure 4.3: Tetrahydral formation

4.2.1 Eclipse Effect

The attitude of the satellites is determined precisely by sun tracking sensors for the subject mission. Given limitations in the accuracy of attitude determination through other methods in the eclipse phase of the orbit the mission specifications define one third of the orbit as unsafe to issue propulsion within accurate specified orientation. The controller is thus required to ensue guidance only in the two thirds of every orbit that attitude can be precisely calculated from the sun.

4.3 Special characteristics of the MPC implementation

Concerning the ability to handle constraints, precisely adjust performance to time-varying mission criteria and further tune any parameters easily after deployment the MPC control strategy is chosen to be distinctively fitting.

Concepts and theory behind MPC can be traced as early as the 1960s mostly in the work of Kalman and the linear quadratic regulator mathematic formulation. As soon as the needs of the industry for system constraints, non-linearities and regular updating were met within feasible computational workloads the use of MPC started to spread starting from the control of chemical and petroleum slow dynamic systems. In the last decades there has been a wide integration of MPC control in industrial applications thanks to the leaps in the computational performance of embedded systems being able to solve their way through demanding quadratic programming problems online for increasingly fast and complex systems.

In the particular three dimensional formation acquisition feat for low earth orbits which involve periodic motion with frequencies in the manner of hours and limited perturbations, the computational resources and power consumption of using a variant of model predictive control are low enough to take the main lead for the ideal control strategy considering the benefits.

Limited perturbations in the current mission are not sufficient to diverge the trajectory or cause any significant change in optimal fuel consumption calculations for a single orbit but for more orbits having the ability to get input data that could be generated and transmitted from Earth as well would degrade the performance too much. For this reason the CubeSats have to be fitted with subsystems that can handle the relatively computationally and energy demanding task of formation acquisition and formation keeping control.

The UWE-4 mission satellites will be equipped with low-thrust electrical propulsion systems that will be responsible to acquire the required orbits, do station-keeping and also converge on the required formations. The goal to acquire these dynamic reference topologies for each satellite with a relatively high accuracy is set to be in the magnitude of days. Such feat for the given satellite altitude and low power propulsion would require planning and control horizons expanding on the scale of hundreds of orbits if implemented in the usual MPC manner.

Combined with the above fact another characteristic that displaces such controller out of traditional usage is the optimal sinusoid like form of the required thrust generated to converge to such topologies and the detailed requirements having to be taken account of within the orbit. A long sampling time control as usually encountered in slow dynamic systems is not fit for such task and cannot produce a detailed thrust function within a single orbit.Furthermore the nature of the satellite mission itself has specific requirements over which parts of the orbit need to have specific constraints and take account formed couplings in a detailed way on the scale of minutes to be able to produce a truly optimal thrust that takes them into account.

Therefore having this small sampling time and a planning and control horizon in the scale of days results in a control problem that is not practically feasible to compute if encountered as in typical MPC implementations. The computational resources and memory usage for such task not only exceed the specifications of the on-board hardware but is nearly impossible to run on a typical enterprise server. For the same reason, any explicit MPC formulation of the problem is out of consideration due to the large memory and power requirements of such implementation and the rigidity of the defined tuning in the span of the mission.

A solution that is feasible within hardware requirements is a control formulation that uses greatly shortened planning and control horizons. The optimized trajectories generated have to converge to the given reference state without the control and planning horizons able to actually extent enough to gain a relatively high proximity to that state for a given iteration, as is usual in the final steps of typical MPC scenarios. This complication for the given task at hand results in the great emphasis that needs to be placed in the balance configuration between various weighted objectives as any miscalculations can lead to problems. The control strategy has to successfully pinpoint a perfect balance between the various state and input variables in its own optimization formulation as any negligence and miscalling can lead to steady state error or even exponential divergence from the given objective.

Additionally, each satellite is controlled to a propagated and not a constant reference state belonging to its specified topology. The low-thrust propulsion of the spacecraft which allows only slight changes in the orbital motion generates trajectory that diverges little from being periodic. Given this specification and the nature of the mission requiring multiple MPC iterations that can arbitrarily vary in their execution phase throughout an orbit, a constant target reference point does not lead to feasible control. For systems as such the re-initiation of a MPC control iteration has to take into account a target reference state equally propagated in its periodic motion as the time it takes to run the next MPC iteration. The reason that the above method is necessary, is that issuing a control trajectory for a constant reference point within the desired formation structure iteratively from different phases of a close to periodic motion would result in opposing thrust profiles and trajectories that do not necessarily accumulate to a common converge in the desired constant reference state.

4.3.1 Decentralized Multistep online MPC formulation

The Decentralized[1] Multistep MPC strategy is followed to reduce the computational load and power consumption of the controllers. The choice for this particular formulation is made taking account the long computationally dormant periods in the orbit like the eclipse phase that could fit a long MPC iteration and the flexibility to allow other computational demanding tasks to be performed in the same processing system without the need to form complex scheduling or to take in mind extensive function calling overhead.

Sensitivity-based multistep MPC and Updated multistep MPC strategies [PSK15] for this particular control implementation are considered redundant due to the extremely low unmeasured perturbations existing in earth orbit and the great accuracy of the models used for orbital system dynamics. A single orbit is considered sufficient to include any on-demand specifications set online during the span of the mission and additionally the derived multistep control input for a single orbit calculated from a single MPC iteration is considered valid within margin of navigational error given the existing plant perturbations.

[1]Decentralized because in the current implementation there is no necessary negotiation between the satellites and the decision process for the target topology states involves exchanging information before and after an MPC iteration even if the reference states are chosen optimally according to the general constellation topology.

Focus is greatly emphasized in the efficient use of fuel,the ability to convergence with high accuracy within short time periods for the given fuel consumption, flexibility to update possible future control criteria throughout an orbit inducing time-variant constraints and weights and the low overall computational load for the MPC iterations.

The controller needs to achieve a perfect balance between having a thrust profile that receives high values in highly efficient parts of the periodic motion without the penalty of non convergence being able to issue extension of such zones to non efficient areas while at the same time acquire a proximity to the target topology with high accuracy after a number of iterations.

4.3.2 MPC Parameter Selection

Scale and constraints of LVLH motion and thrust

The scale of the mission has to be accurately defined as it is part the balance mechanism pinpointing the importance between generated thrust inputs and convergence:

Magnitude:	Thrust	Position	Velocity
X axis		2km	2m/s
y axis		1km	1m/s
Z axis		1km	1m/s

Propulsion constraints: 8 μN low thrust propulsion systems and a mass of approximately 1.3 kg.

Thrust constraints have to be manually set to zero for the steps after the control horizon ends until the end of the planning horizon because of the default MPC implementation setting the value of the final MV input in the control horizon to remain constant throughout the calculation of the optimal trajectory, which is not the case for this system. [2]

Variable Weights

[2]The system's LVLH motion does not need to be constrained in the current mission objectives.

The thrust input has constant weights as normally encountered in typical MPC implementations but the state divergence output is set to have terminal weights over constant weights or time-varying arbitrary weights because of the control objective being its final state convergence. The final state has to be set distinctively as the most important factor as any intermediate states can cause redundant trajectory penalty and increased fuel consumption having little effect in the mission [3].

Thrust input (MV) weight: 1

State divergence (M0) weight: 1.7

For every new objective the ratio between the thrust input and state divergence weights has to be changed accordingly to fit the required convergence timespan and fuel efficiency criteria. Any variable weight in any of the given axes can be further adjusted given more specific criteria[4].

Control and Planning Horizons

The resulting feat, to be feasible, involves a controller able to converge the given state of a spacecraft to a dynamic target state that is part of the required formation topology without necessarily being able to acquire a proximity to it within the planning horizon. Thus the planning and control horizons used are the following:

Control horizon adjusted to approximately two thirds of an orbit. Within that span various possible online mission specifications that require change of different weights and constraints throughout an orbit can be gratuitously enforced and a Multistep MPC iteration can generate a sufficient number of thrust vector inputs within the duration of an ellipse.

Planning horizon adjusted to be two orbits. Reference target state needs to be approximated within a number of timesteps ahead of the control horizon to allow the force induced propagation of the satellites to accumulate in order to give a good estimation of the related convergence and fuel usage penalties.

[3]The weights are taking the same value uniformly through their corresponding variables as the scaling factors are setting an estimation of the proximity and importance of the given raw numbers.

[4]The rates themselves have no meaning for the accumulated cost and should not effect the optimization problem.

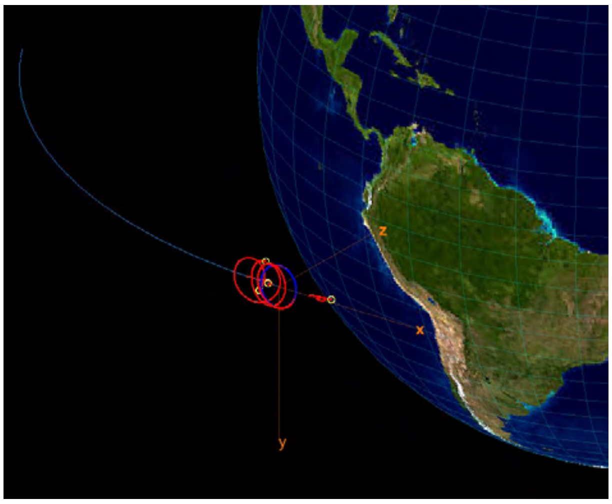


Figure 4.4: a

Sampling time

The sampling time has to be small enough to produce a detailed thrust function and approximate online in-orbit requirements but also big enough to reduce the planning and control horizons spanning in a manner of orbits to a number of steps computationally and resource feasible within the onboard hardware.

Sampling time: 240 seconds

Control Horizon steps: 18

Planning Horizon steps: 33

4.4 Simulation

For the evaluation of the controller the following formation acquisition maneuver is used which represents a real mission objective in the NetSat mission: Acquisition from Tetrahedral to ”1 in plane-3 out of plane” formation, demonstrated in the following figures.

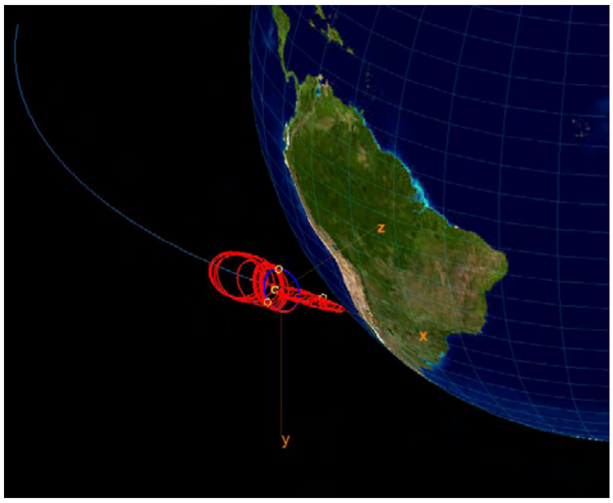


Figure 4.5: b

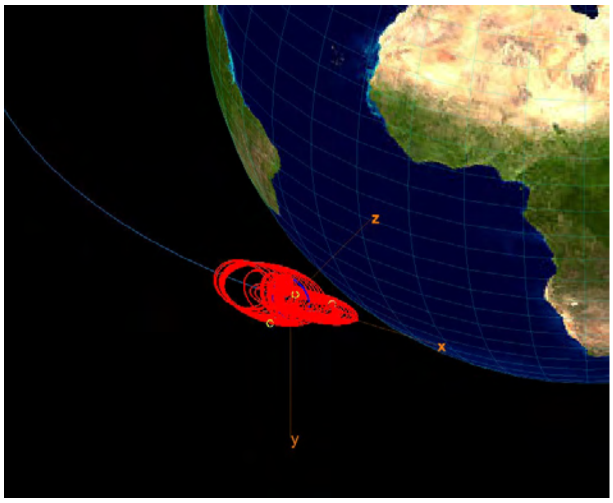


Figure 4.6: c

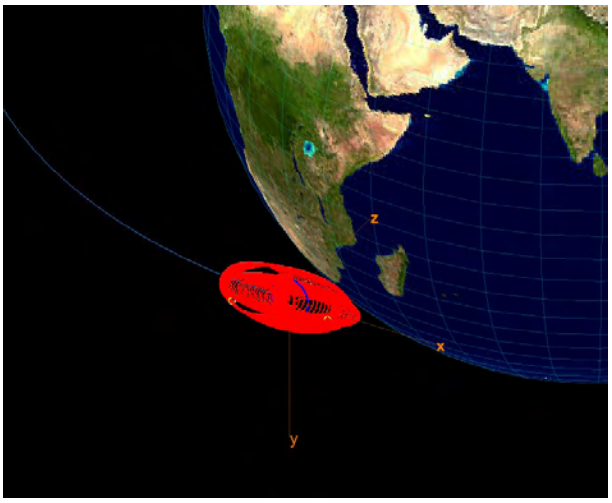


Figure 4.7: d

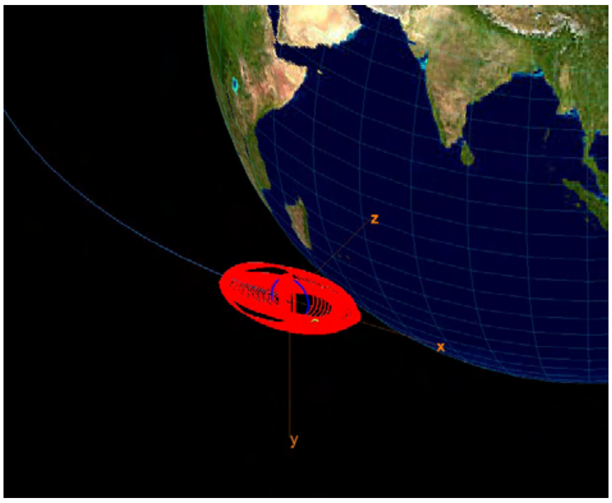


Figure 4.8: e

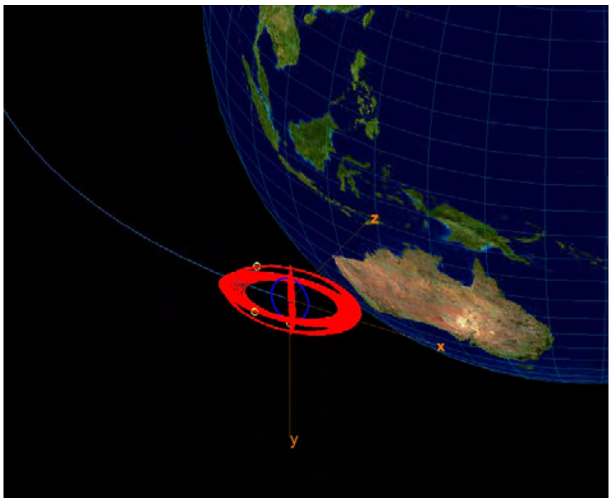


Figure 4.9: f

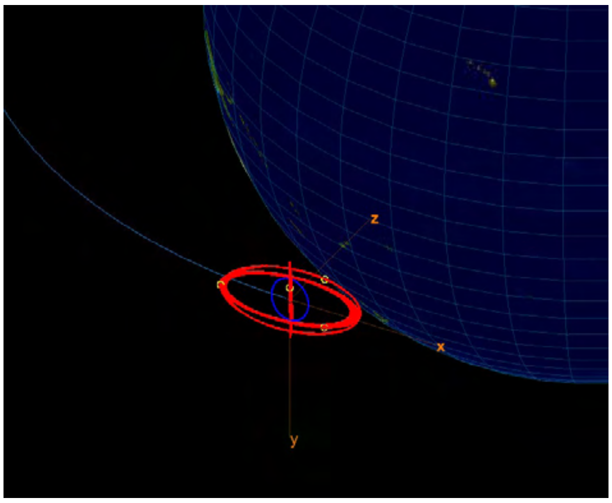


Figure 4.10: g

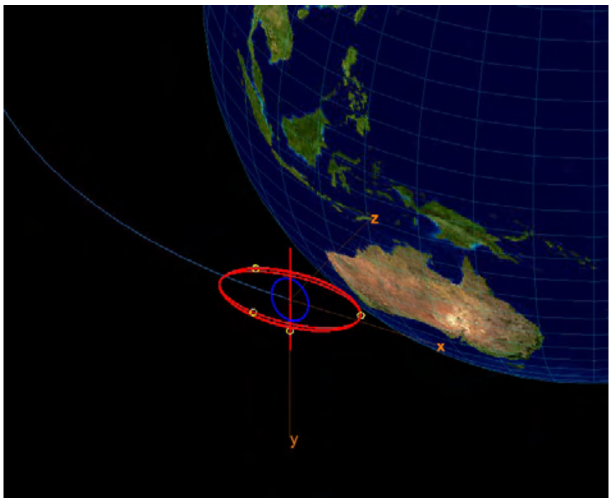


Figure 4.11: h

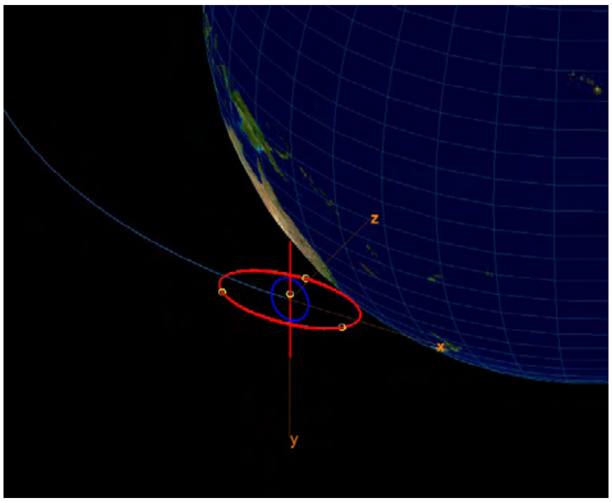


Figure 4.12: i

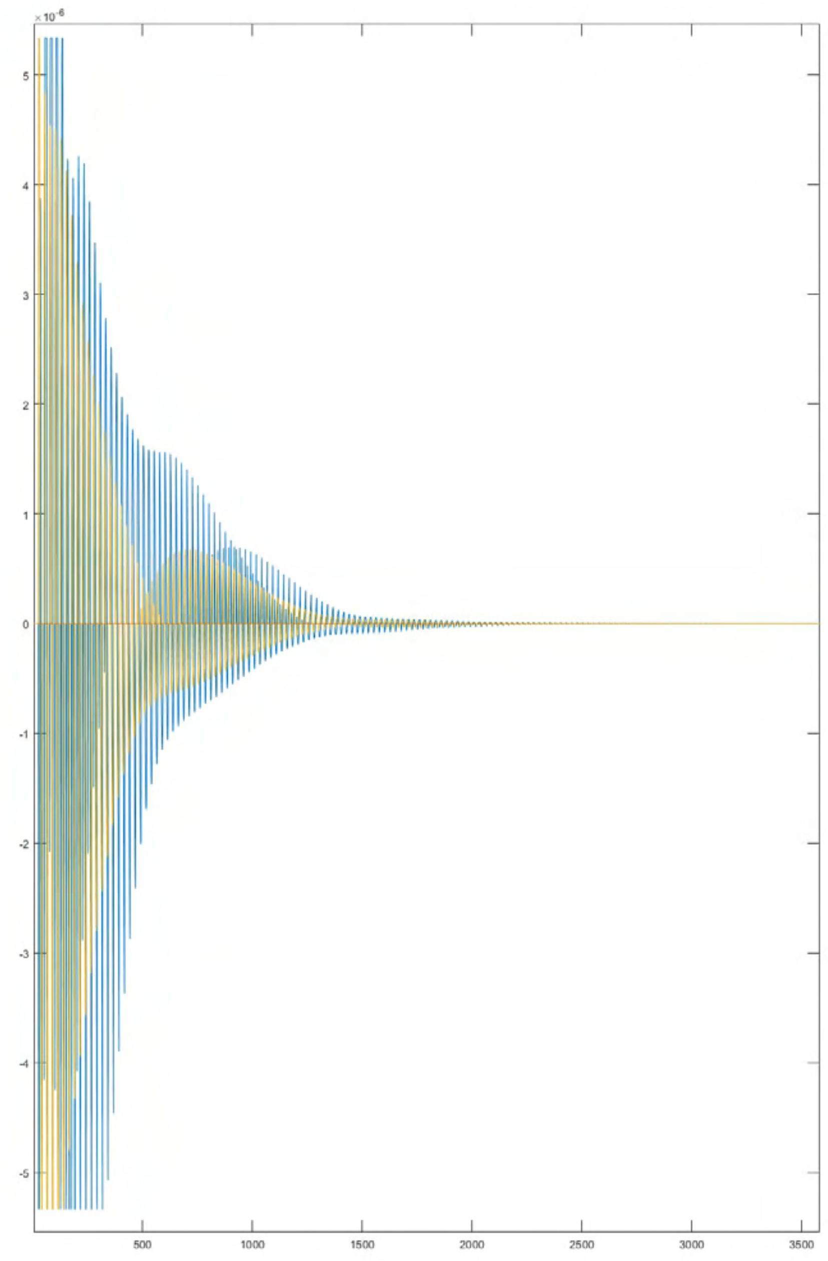


Figure 4.13: Thrust function generated for the acquisition

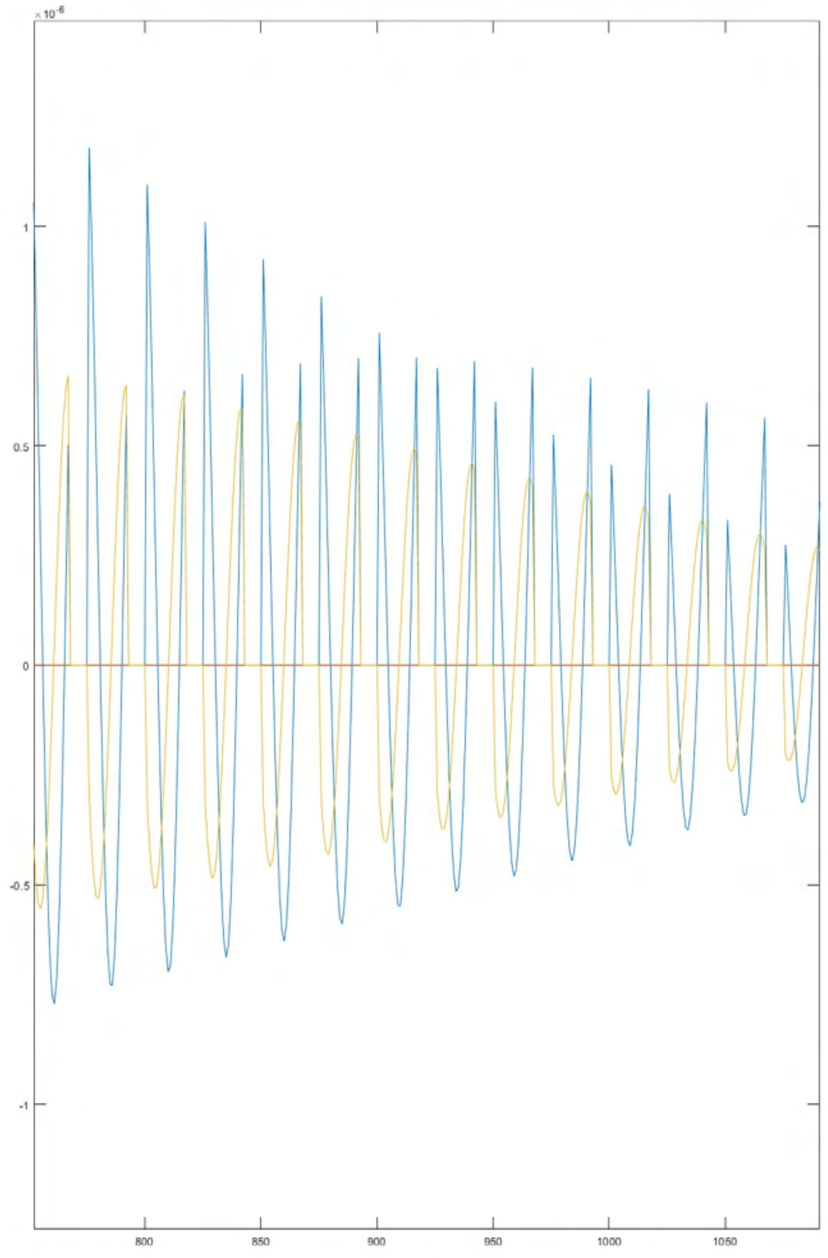


Figure 4.14: Thrust function, detailed view

4.5 Conclusion

This paper proposes MPC as a suitable control strategy for continuous low thrust formation flying in LEO and defines a set of parameters for MPC that both leads to convergence to a given target formation and is can be realistically performed on embedded hardware as met on nowadays small satellites. Simulations using Matlab have been performed to show the performance of the described MPC.

MPC proves advantageous due to several reasons, namely its optimality, its ability to take various constraints into account and most – which is most important for low thrust applications – its ability to plan for a distinct time into the future.

This controller development is planned to be implemented on the described NetSat mission. This mission can also act as in-orbit verification of the method presented.

Chapter 5

Code section

Main Function:

```
%%Panayiotis D. Kremmydas
%Multi−Sattelite Formation control, Simulated with respect to Earth
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%X,Y,Z values correspond to the LVLH Coordinate system as defined by ESA, (units set in meters)

%%%%%%User input:
addpath('lvlh_formation_trajectory_generators'); %dir: formations fucntion file



%% Orbit and Formation variables
    EarthMass = 3.986004415e14; % Earth mass in m^3
    EarthRadius = 6371000; %Mean Earth radius in m
    orbitalRadius= EarthRadius + 700000; % for circular orbit with altitude 700km
angularSpeed=sqrt(EarthMass /orbitalRadius^3); % corresponding to force free circular orbit: sqrt(3.986004415e14/(6371000+700000)^3)=0.001061812,
    [1/s] ~Example: ISS's angular frequency
    0.00113027258 in rads/second, Altitude=400000,
    Orbital period=92.65 minutes=5559seconds, %sqrt
    Precision might not be sufficient
T=2*pi/angularSpeed;  %5917seconds or 100minutes. %
    Altitude matching a specific circular angular
    frequency and period of orbit around an object of a
     specific mass
%Formation structure parameters
semi_major_xz_plane_ellipse=400; %semi-major of the
    formation structure
init_y_chief_amplitude=150;  %LVLH y periodic
    amplitude of the formation structure
satellite_number=4; %Number of satellites for the
    given formation structure. {Tetrahydral formations
    are only supported by 4 satellites}
tetrahydral_binary_configuration=1; % Tetrahydral
    formation parameter: orientation of the surface
    formed by the 2 LVLH non-stationary satellites, 1=
    pointing to the north, -1=pointing to the south
%Simulation Parameters
free_drift_orbital_revolutions=2; %initial state
    orbital revolutions to be simulated
transfertime_orbital_revolutions=100; %controlled
    state orbital revolutions to be simulated
propagate__orbital_revolutions=70; %propagated state
    orbital revolutions to be simulated
formation1=4; %initial freedrift formation    %
    cw_moving_plane=1,cw_pco=2,cw_plane_1out=3,
    cw_equidistant=4,tetrahedral=5,cw_helix=6
formation2=6; %taget formation to be acquired %
    cw_moving_plane=1,cw_pco=2,cw_plane_1out=3,
    cw_equidistant=4,tetrahedral=5,cw_helix=6
sampleInterval=80; %every sampleInterval seconds a
    sample is calculated and processed into the
    simulation.
simulationStyle=1; %SPACE BACKGROUND=1 %Smooth orbit
    transition=2 %without background stars simulations
    are faster
%Variables%%%%%%%
t_freedrift = 0:sampleInterval:
    free_drift_orbital_revolutions*T;
t1_len=length(t_freedrift);
t_transfer  = 1:1:transfertime_orbital_revolutions*(
    ceil(T/sampleInterval));
t2_len=length(t_transfer);
t_propagate= 0:sampleInterval:
    propagate__orbital_revolutions*T;
t3_len=length(t_propagate);
total_samples=t1_len+t2_len+t3_len;

%% Examples of force free formations  (where s the
    sample amount and k the number of satellites)
%function  [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k
    ,1:s),dz(k,1:s)]= cw_equidistant(
    semi_major_xz_plane_ellipse,satellite_number,
    angularSpeed,t);
%%%%%%CW Equidistant Formation: x0=arbitrary, y0=0 ,
    z0=0 , dx0=0 , dy0=x0*sqrt(0.75)*omega , dz0=x0
    *(0.5)*omega
%function  [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k
    ,1:s),dz(k,1:s)]= cw_pco(
    semi_major_xz_plane_ellipse,satellite_number,
    angularSpeed,t);
% %%%%%Projected Circular Orbit (on X-Y plane): x0=
    arbitrary, y0=arbitrary , z0=0 , dx0=0 , dy0=-2dz0
    , dz0=(0.5)*omega*x0
%function  [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k
    ,1:s),dz(k,1:s)]= cw_helix(
    semi_major_xz_plane_ellipse,satellite_number,
    angularSpeed,t);
% %%%%%CW Helix Formation: chief: x0=arbitrary, y0=x0
    /2 , z0=0 , dx0=0 , dy0=0 , dz0=x0*(0.5)*omega
%function  [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k
    ,1:s),dz(k,1:s)]= cw_moving_plane(init_y_amplitude,
    semi_major_xz_plane_ellipse,satellite_number,
    angularSpeed,t);
% %%%%%CW Moving Plane Formation (Moving Y bar on YZ
    and YX axes): x0=(+-k)*arbitrary, y0=arbitrary , z0
    =0 , dx0=0 , dy0=arbitrary , dz0=(+-k)*(0.5)*omega*
    x0,  where k is the corresponding number of the
    satellite
```

```
%function  [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k
    ,1:s),dz(k,1:s)]= cw_plane_1out(
    init_y_chief_amplitude,semi_major_xz_plane_ellipse,
    satellite_number,angularSpeed,t);
% %%%%%CW one plane formation , 1 out of plane: x0=
    arbitrary, y0=arbitrary , z0=0 , dx0=0 , dy0=-2dz0
    , dz0=(0.5)*omega*x0
%function  [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k
    ,1:s),dz(k,1:s)]= tetrahedral(
    tetrahydral_binary_configuration,angularSpeed,t);
% %%%%%Tetrahedral Formation (constant volume inside
    formation): chief: x0=arbitrary, y0=x0/2 , z0=0 ,
    dx0=0 , dy0=0 , dz0=x0*(0.5)*omega
%Output has this form [x(k,1:s),y(k,1:s),z(k,1:s),dx(k
    ,1:s),dy(k,1:s),dz(k,1:s)]


%% 1: Formation FORCE FREE Initial State %(k,0:t1_len)

    if      formation1==1          %cw_moving_plane
                [xtot,ytot,ztot,dxtot,dytot,dztot]=   ...
            cw_moving_plane(semi_major_xz_plane_ellipse,
               satellite_number,angularSpeed,t_freedrift
               );
    elseif  formation1==2          %cw_pco
                [xtot,ytot,ztot,dxtot,dytot,dztot]=   ...
            cw_pco(semi_major_xz_plane_ellipse,
               satellite_number,angularSpeed,t_freedrift
               );
    elseif  formation1==3          %cw_plane_1out
                 [xtot,ytot,ztot,dxtot,dytot,dztot]=
                    ...
             cw_plane_1out(init_y_chief_amplitude,
                semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,
                t_freedrift);
    elseif  formation1==4          %cw_equidistant
                 [xtot,ytot,ztot,dxtot,dytot,dztot]=
                    ...
             cw_equidistant(semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,
                t_freedrift);
    elseif formation1==5          %tetrahedral
                    [xtot,ytot,ztot,dxtot,dytot,dztot]=
                        ...
               tetrahedral(semi_major_xz_plane_ellipse,
                   tetrahydral_binary_configuration,
                   angularSpeed,t_freedrift);
    elseif formation1==6          %cw_helix
                    [xtot,ytot,ztot,dxtot,dytot,dztot]=
                        ...
               cw_helix(semi_major_xz_plane_ellipse,
                   satellite_number,angularSpeed,
                   t_freedrift);
    end

%% 2: Formation CONTROL Acquisition Transfer
    if      formation2==1          %cw_moving_plane
                    [x1,y1,z1,dx1,dy1,dz1]=   ...
            cw_moving_plane(semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,0);
    elseif formation2==2          %cw_pco
                    [x1,y1,z1,dx1,dy1,dz1]=   ...
            cw_pco(semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,0);
    elseif formation2==3          %cw_plane_1out
                    [x1,y1,z1,dx1,dy1,dz1]=   ...
              cw_plane_1out(init_y_chief_amplitude,
                  semi_major_xz_plane_ellipse,
                  satellite_number,angularSpeed,0);
    elseif formation2==4          %cw_equidistant
                    [x1,y1,z1,dx1,dy1,dz1]=   ...
              cw_equidistant(semi_major_xz_plane_ellipse,
                  satellite_number,angularSpeed,0);
    elseif formation2==5          %tetrahedral
                    [x1,y1,z1,dx1,dy1,dz1]=   ...
              tetrahedral(semi_major_xz_plane_ellipse,
                  tetrahydral_binary_configuration,
                  angularSpeed,0);
    elseif formation2==6          %cw_helix
                    [x1,y1,z1,dx1,dy1,dz1]=   ...
              cw_helix(semi_major_xz_plane_ellipse,
                    satellite_number,angularSpeed,0);
    end


    for k = 1:satellite_number
        %FreeDriftFormations(x0,y0,z0,dx0,dy0,dz0,omeg,t) values propagate according to the CW equations for formation1 orbits
      [y,rt,u,x,MPCobj,controllerState,Iterations] = mpc_Wrapper_offline(xtot(k,t1_len), ...
                                       ytot(k,t1_len), ...
                                       ztot(k,t1_len), ...
                                       dxtot(k,t1_len), ...
                                       dytot(k,t1_len), ...
                                       dztot(k,t1_len), ...
                                       x1(k), ...
                                       y1(k), ...
                                       z1(k), ...
                                       dx1(k),...
                                       dy1(k),...
                                       dz1(k),...
                                       angularSpeed, sampleInterval,t2_len);
              xtot(k,t1_len+1 : t1_len+t2_len)=y(1,:);
              ytot(k,t1_len+1 : t1_len+t2_len)=y(3,:);
              ztot(k,t1_len+1 : t1_len+t2_len)=y(5,:);
             dxtot(k,t1_len+1 : t1_len+t2_len)=y(2,:);
             dytot(k,t1_len+1 : t1_len+t2_len)=y(4,:);
             dztot(k,t1_len+1 : t1_len+t2_len)=y(6,:);
            sum(sum(abs(((100*u)*(100*u'))*sampleInterval)))
            sum(sum(abs(u*sampleInterval)))
            %%%%%%%%%%%%%%%%
             display('satellite :////////////////////////////////////////////////////////', num2str(k))
%Extra plotting options and variable storage { Necessary to crosscheck embedded systems implementation}
%           if (k==3)
%               objectTrajectory=y;
%                   objectReference=rt;
%                   objectOptimalThrust=u';
%                   objectControllerState=controllerState;
%                   objectQP_Iterations=Iterations;
%                   objectInitialState=x;
%                    figure
%                     plot(objectOptimalThrust)
%                     return;
%              end
           %%%%%%%%%%%%%%%%%%%
    end

%% 3: Formation FORCE FREE Propagated States   %final state values of the MPC output, propagated in the Hill equations
    for k = 1:satellite_number
            [xtot(k,t1_len+t2_len+1:total_samples), ...
             ytot(k,t1_len+t2_len+1:total_samples), ...
             ztot(k,t1_len+t2_len+1:total_samples), ...
            dxtot(k,t1_len+t2_len+1:total_samples), ...
            dytot(k,t1_len+t2_len+1:total_samples), ...
            dztot(k,t1_len+t2_len+1:total_samples)]= ...
      FreeDriftFormations(    xtot(k,t1_len+t2_len), ...
                              ytot(k,t1_len+t2_len), ...
                              ztot(k,t1_len+t2_len), ...
                             dxtot(k,t1_len+t2_len), ...
                             dytot(k,t1_len+t2_len), ...
                             dztot(k,t1_len+t2_len), ...
                               angularSpeed, t_propagate);
    end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Coordinate transformation from LVLH (X,Y,Z) to FIGURE
    COORDINATES(X,Z,−Y).
    %Matlab's default viewing limitations in its 3D
       figure representation allow 3D orientation with
        respect to the xy axes only
    % Camera focusing in the perspective of the xy
       axes instead of the xz is not intuitive in the
       current simulation framework
    %  the y and z axes are switched to overcome that
       viewing limitation and the earths's north pole
       is displayed as top (−Y)
    % The figure coordinates are thus used for a more
       intuitive global presentation (X,Z,−Y),
    %  and the local LVLH coordinate system is
       separately displayed by itself within it.

%% Plotting options
close all %closes existing figures when restarting the
    simulation
Simulation=figure('Color', 'k');  %color k sets the
   background of the figure to black.
set(0,'DefaultFigureWindowStyle','docked') %so that
   any subsequent figures are becoming tabs the figure
    window
hold on;
set(gca,'Visible','off', 'NextPlot','add');%Removes
   default axes that do not have any physical for the
   presentation (X,Z,−Y)
axis vis3d; %Locks the aspect ratio of all the axes to
    1:1:1 for accurate 3D model presentation
view(90,0); %Default perspective in case animation
   function: view(59−0.8*k,41+47*sin(0.0055*k+3.1*pi
   /2)) is turned off



%% Plotted objects and their options for: Satellities,
    Initial Satellite trail, Controlled Satellite
   trail,
%Terminated Control Satellite trail, Chief−Center of
```

```
    LVLH, Dummy XZ plane
for k = 1:satellite_number
    sat(k) = animatedline('Color','y','LineWidth',1.5,
       'MaximumNumPoints',1,'Marker','o'); %Current
       location of the Satellite object
    trail(k) = animatedline('Color','b','LineWidth'
       ,1.5,'MaximumNumPoints',6*ceil(20*(50/
       sampleInterval))); %Initial Satellite trail
    trail_with_control(k) = animatedline('Color','r','
       LineWidth',1.5,'MaximumNumPoints',500); %
       Controlled Satellite trail
    trail_release(k) = animatedline('Color','g','
       LineWidth',1.5,'MaximumNumPoints',420); %
       Terminated Control Satellite trail
end
%satchief = animatedline('Color','y','LineWidth',1.5,'
    MaximumNumPoints',1,'Marker','o'); %Chief-Center of
     LVLH
%addpoints(satchief,0,0,0); %Chief-Center of LVLH
%dummy = animatedline('Color','r','LineWidth',1.2,'
    MaximumNumPoints',60,'LineStyle','-.'); Dummy XZ
    plane


%% Plotting options: Rotating Earth  FIGURE
    COORDINATES(X,Z,-Y)
hgx = hgtransform; %superclass of all the objects to
    be transformed around their z axis. (only the earth
     itself currently)

earthsize=6000;     %earth object radius. simulation
    relative quantity only
earthdistance=8000; %earth distance from center of
    formation. simulation relative quantity only
          %ellipsoid(x,y,z, xsize,ysize,zsize, num of
              panels)
%%%%EARTH object: 3D meshgrid using the ellipsoid
    function on a preset number of total points
    defining the above sphere on the predefined
    distance.
[temp1, temp2, temp3] = ellipsoid(0, earthdistance, 0,
    earthsize, earthsize, earthsize, 40);  %less than
    40 makes earth less circular.
globe = surf(temp1, temp2, -temp3, 'Parent',hgx); %%
    Draw the above spherical wireframe globe and add it
     to the hgx class  which will allow object
    transformation operations
data = imread('NasaEarth.jpg'); %  Earth image for
    texture map, must be 2:1 unprojected globe, imread
    transforms it in a CData form
%Setting the meshgrid to have a Texturemap contained
    in "data"  wrapped around it:
set(globe, 'FaceColor', 'texturemap', 'CData', data, '
    FaceAlpha', 1, 'EdgeAlpha', 0.2, 'EdgeColor',
    0.9*[0 1 1]); %Facecolor indicates a texturemap,
    which Matlab expects to be in cdata.


if (simulationStyle==1) %Style1: With stars %%%%SPACE
    BACKGROUND object: 3D meshgrid using the ellipsoid
    function on a preset number of total points
    defining the above sphere on the predefined
    distance.
    [temp1, temp2, temp3] = ellipsoid(0, 0, 0, 60000,
       60000, 60000, 40);
    globe = surf(temp1, temp2, -temp3); %% Draw the
       above spherical wireframe sphere and add it to
       the hgx class  which will allow object
       transformation operations
    data = imread('NasaStarmap.jpg'); %  Star image
       for texture map, must be 2:1 unprojected globe,
        imread transforms it in a CData form
   %Setting the meshgrid to have a Texturemap
       contained in "data"  wrapped around it:
    set(globe, 'FaceColor', 'texturemap', 'CData',
       data, 'FaceAlpha', 0.7, 'EdgeAlpha', 0, '
       EdgeColor', 0*[0 1 1]); %Facecolor indicates a
       texturemap, which Matlab expects to be in cdata
       .
    camzoom(1);
else
    camzoom(2.6);
```

```
end

%% Plotting options: 3 axes and velocity tracers for the Satellite Formation: FIGURE COORDINATES(X,Z,-Y)
plot(-8000*sin(0:0.05:0.8*pi),8000-8000*cos(0:0.05:0.8*pi)); %velocity tracers
axes_size=0:(earthdistance-earthsize)/1.4;
plot3(axes_size,zeros(1,numel(axes_size)),zeros(1,numel(axes_size)),'Color',[1 0.5 0 0.3]);
plot3(zeros(1,numel(axes_size)),axes_size,zeros(1,numel(axes_size)),'Color',[1 0.5 0 0.3]);
plot3(zeros(1,numel(axes_size)),zeros(1,numel(axes_size)),-axes_size,'Color',[1 0.5 0 0.3]);
text(numel(axes_size),0,0, '\bfx', 'Rotation',+0,'Color',[1 0.5 0 0.01],'FontSize',14);
text(0,numel(axes_size),0, '\bfz', 'Rotation',+0,'Color',[1 0.5 0 0.01],'FontSize',14);
text(0,0,-numel(axes_size), 'y', 'Rotation',+0,'Color',[1 0.5 0 0.01],'FontSize',14);

xlim([-2*earthdistance 2*earthdistance])
ylim([-2*earthdistance 2*earthdistance])
zlim([-2*earthdistance 2*earthdistance])


%% Simulation: Draw Satellites, Rotate Globe, Handle Figure: CONVERTED TO FIGURE COORDINATES(X,Z,-Y)
set(gcf,'Position', [0 0 1300 800]); %custom window size, custom simulation size
set(gcf,'CurrentCharacter','@');
for k = 1:length(xtot)
    %figure(Simulation);  %when implementing hotkeys its important to keep focus on the figure.
    for i = 1:satellite_number
        %%%%%%%%%%%%
        if(k<t1_len)
            addpoints(trail(i),xtot(i,k),ztot(i,k),-ytot(i,k)); %does allow fading normal trail
        elseif  (k<t1_len+t2_len)
            addpoints(trail_with_control(i),xtot(i,k),
            ztot(i,k),-ytot(i,k)); %does allow fading control trail
        else
            addpoints(trail_release(i),xtot(i,k),ztot(i,k),-ytot(i,k));
        end
        addpoints(sat(i),xtot(i,k),ztot(i,k),-ytot(i,k));
        %%%%%%%%%%

%           addpoints(trail(i),xtot(i,k),ztot(i,k),-ytot(i,k));
%           addpoints(sat(i),xtot(i,k),ztot(i,k),-ytot(i,k));
    end
    %addpoints(dummy,x(1,k),z(1,k),0); %Draw the XZ ellipse for reference {corresponding to a formation without any y displacement or velocity)


    set(hgx,'Matrix', makehgtform('translate',[0,8000,0],'zrotate',-(sampleInterval/T)*k*2*pi,'translate',[0,-8000,0])) %Rotating Globe; Rotation= T/sampleInterval


    %%%%%%%%Drawing and Camera motion styles:
    if (simulationStyle==1)%Style1: With stars
        distanceConstant=25000;
        xlim([distanceConstant*cos(0.045*k)-distanceConstant-14000 distanceConstant*cos(0.045*k)+distanceConstant+14000])
        ylim([distanceConstant*sin(0.045*k)-distanceConstant-14000 distanceConstant*sin(0.045*k)+distanceConstant+14000])
        zlim([-3*distanceConstant 3*distanceConstant])
        drawnow
        campos([-2*distanceConstant*cos(0.045*k) -2*distanceConstant*sin(0.045*k) 2*distanceConstant+distanceConstant*sin
                (0.0045*k)])
            camtarget([0,0,0])
        else %Style2: Smooth orbit transition
            view(159-0.8*k,41+47*sin(0.0045*k+3.1*pi/2));
                %Handle Camera
            drawnow
        end

        %if mod(k,100)==0, pause; end                    %
            Image Capturing and exportation
        %if mod(k,100)==0, disp(k*100/length(xtot));     %
            Simulation progress display

        %inLoopTakeFrame_AppendToGif(k);                 %
            Simulation capturing as .gif


end


%         key=get(gcf,'CurrentCharacter');
%         if key~='@' % if it changed from the dummy
    character
%             set(gcf,'CurrentCharacter','@'); % reset the
     character
%            % now process the key as required
%            if key=='q'
%                break;
%            %elseif key<=6 && key>=1
%            %     newformationacquisition(key);
%            end
%        end
```

Multistep Model predictive control Wrapper function

```
%mpc_Wrapper Author:   Panayiotis D. Kremmydas
%Reference: HCW_MPC_Example_2:    Julian Scharnagl

```

```
%Multistep MPC based on
%Co-ordination and control of distributed spacecraft
%systems using convex optimization techniques
%   Detailed explanation goes here
    %start mean [a theta i q1 q2 Omega lambda e]
    %final time (offset from now) Tf [s]
    % target vector at Tf: Xfinal
    %time step Ts (integration step size)

function [y,r,u,x,MPCobj,controllerState,iterations] =
    mpc_Wrapper_offline(x0,y0,z0, dx0,dy0,dz0,x1,y1,z1
   ,dx1,dy1,dz1, w, tstep,T)%the controller needs to
   output the predicted states and controls within the
     controlHorizon
%% Definition of constraints {User Input}, implemented
    with #define in embedded C
planningHorizon = ceil((2*pi/w)/tstep); %
   planningHorizon=multiple of Number of steps per
   orbit
s0 = [x0; dx0; y0; dy0; z0; dz0];
sr = [x1; dx1; y1; dy1; z1; dz1];


[MPCobj,statespace]= netSATmpc(w, tstep);
 [rtemp,~,~] = lsim(statespace,zeros(3,2*T), 0:tstep:
    tstep*(2*T-1),sr); %%using the state space itself
    to propagate th reference state
 r =rtemp'; %r=transpose(zeros(1,6));

% [previous_reference_state,~,~] = lsim(statespace,
   zeros(3,2*T), tstep*(steps_since_last_mpc_call),
   previous_reference_state); %%using the state space
   itself to propagate th reference state
%steps_since_last_mpc_call includes the number of
   steps that the current satellite state has been
   propagated (e.g. steps propagated using the MPC
   inputs or steps propagated using zero inputs in the
    case of the
%eclipse)

x = mpcstate(MPCobj);  %Current state
```

```
y(:,1) = s0;            %Output state y
x.Plant= s0;            %internal state of the mpc
   controller needs to be initiallized.
x.LastMove= [0 0 0];   %internal state of the mpc
   controller needs to be initiallized.
x.Disturbance= [0 0 0 0 0]';
x.Covariance= zeros(11,11);
controllerState = s0;
iterations=[];
%u = zeros(3, T);        %Preallocating memory for
   faster simullations    %Input thrust u

options = mpcmoveopt; %Like allocating weights on
   runtime
% options2 = mpcsimopt(); %used for sim
% options2.PlantInitialState=s0;
% %options2.OutputNoise = s0;
% options2.RefLookAhead = 'on';
% options2.MDLookAhead = 'on';
% options2.Constraints = 'on';
% options2.OpenLoop = 'off';
% sim(MPCobj, T, rtemp,[],options2);
% pause
offline_steps=ceil((2*pi/w)/tstep);%planningHorizon-3
   %planningHorizon; %planningHorizon;
for i = 2:offline_steps:T+offline_steps %T+
   offline_steps needs to calculate a few extra steps
   but only output the ones necessary y=y(:,2:T+1);


 %  needs to be readjusted to the correct reference
    state given at the end of the planning horizon
        disp(['i: ', num2str(i), ' of a total of ',
           num2str(T), ...
           ' and current error: ', num2str(norm(abs(y
              (1:2:5,i-1) ...
           - rtemp(i-1,1:2:5)')))]);
 %  end

    [unused,mpcmoveInfo]= mpcmove(MPCobj, x, x.Plant,
       rtemp(i,:)  , [], options); %[0 0 0 0 0 0],
        options); %y3(:,i:i+offline_steps -1)=
        mpcmoveInfo.Yopt(1:offline_steps ,:)' not%
        working %rtemp(i+ceil((1*pi/w)/tstep) for phase
         shift in the target reference
    %u(1:3,i:i+offline_steps -1) = mpcmoveInfo.Uopt(1:
        offline_steps ,:)'; %thrust profile

    %Including Eclipse effect. Recommended use of a
        sampling time lower than 120
    for k=0:offline_steps -1
        if k< ceil((2/3)*(2*pi/w)/tstep)  %mod((k+i)*
            tstep,(2*pi/w))< (2/3)*(2*pi/w)
             y_move = lsim(statespace,[mpcmoveInfo.Uopt
                (k+1,:) ; zeros(1,3)],0:tstep:tstep,y
                (:,i+k-1));
             y(:,i+k) = y_move(2,:);
             u(1:3,i+k)=mpcmoveInfo.Uopt(k+1,:);
        else
            y_move = lsim(statespace,[zeros(1,3) ;
               zeros(1,3)],0:tstep:tstep,y(:,i+k-1));
            y(:,i+k) = y_move(2,:);
            u(1:3,i+k)=zeros(1,3);
        end
    end

     %Without Eclipse effect.
%      for k=0:offline_steps -1
%              y_move = lsim(statespace,[mpcmoveInfo.
    Uopt(k+1,:) ; zeros(1,3)],0:tstep:tstep,y(:,i+k-1))
    ;
%              y(:,i+k) = y_move(2,:);
%              u(1:3,i+k)=mpcmoveInfo.Uopt(k+1,:);
%      end

    x.Plant= y(:,i+offline_steps -1); %internal state
        of the mpc controller needs to be updated
    x.LastMove= mpcmoveInfo.Uopt(offline_steps ,:); %
        zeros(1,3);
   % x.Disturbance= [0 0 0 0 0]';
   % x.Covariance= zeros(11,11);
    controllerState= [controllerState x.Plant];
    iterations=[iterations mpcmoveInfo.Iterations];
        for k=0:1:offline_steps
         u(3*k+1:3*k+3)=mpcmoveInfo.Uopt(k+1,:);
        end

 %       mpcmoveInfo.Cost
%       if max(abs(y(:,i-1) - r(:,i-1)).*[1;100;1;100;1;100]) < 0.1
%           if max(mpcmoveInfo.Cost) < 10
%               display('Time in days')
%             time=(i+offline_steps)*tstep/(60*60*24)
%             squarecost=sum(sum(abs(((100*u)*(100*u'))*tstep)))
%             abscost=sum(sum(abs(u*tstep)))
%             plot(u')
%             return;
%     end
end

u=u(:,2:T+1);
y=y(:,2:T+1);
```

MPC configuration function

```
%netSATmpc Author:  Panayiotis D. Kremmydas
%Reference: HCW_MPC_Example_2:   Julian Scharnagl
%MPC object and state space definition function

function [MPCobj,sys1] = netSATmpc(w, tstep)

%Including Eclipse effect. Recommended use of a sampling time lower than 120
planningHorizon = ceil(2*(2/3)*(2*pi/w)/tstep) %Has to be a multiple of the total number of steps an orbit
controlHorizon = ceil((2/3)*(2*pi/w)/tstep)+1  %Mitigates the effort to reach referrence into more steps. Should be equal to planningHorizon for thrust profiles

```

```
%Without Eclipse effect.
% planningHorizon = ceil(3*(2*pi/w)/tstep) %Has to be a multiple of the total number of steps an orbit
% controlHorizon = ceil((2*pi/w)/tstep)+2  %Mitigates the effort to reach referrence into more steps. Should be equal to planningHorizon for thrust profiles

%2 * stepsOrbit; % Control horizon in steps is 2 orbits. Control horizon can be anything below planning horizon. %%%%%%%%
%For the implementation of satellites it makes sense to have a control horizon and a planning horizon only when the reference state is propagated
%when it is not having a controller that achieves a steady state earlier than the acquisition time would result in a a trajectory that
%loops around the acquired state until the end of the manuever trying to hold that steady state , resulting in major cost in fuel consumption.
%% Contineus State space model of the Hill equations

A = [    0,        1,        0,        0,        0,        0;
         0,        0,        0,        0,        0,        2*w;
         0,        0,        0,        1,        0,        0;
         0,        0,        -w*w,     0,        0,        0;
         0,        0,        0,        0,        0,        1;
         0,        -2*w,     0,        0,        3*w*w,    0];

B =   [  0,        0,        0;
         1,        0,        0;
         0,        0,        0;
         0,        1,        0;
         0,        0,        0;
         0,        0,        1];

C = eye(6);

D = zeros(6,3);


```

```

sys1 = ss(A,B,C,D);
sys1.InputGroup.ManipulatedVariables = 1:3;
sys1.OutputGroup.MeasuredOutputs = 1:6;

sys1=c2d(sys1,tstep);


verbosity = mpcverbosity('off'); % Temporarily disable command line messages.
MPCobj= mpc(sys1, tstep,planningHorizon,controlHorizon);
verbosity = mpcverbosity('on');

  maxAcc = 8e-6/1.5;  %3* 5.3N
%       MPCobj.ManipulatedVariables(1).Min = -maxAcc;
%       MPCobj.ManipulatedVariables(1).Max = +maxAcc;
%       MPCobj.ManipulatedVariables(2).Min = -maxAcc;
%       MPCobj.ManipulatedVariables(2).Max = +maxAcc;
%       MPCobj.ManipulatedVariables(3).Min = -maxAcc;
%       MPCobj.ManipulatedVariables(3).Max = +maxAcc;
          %Thrust has to be manually set to zero for the remaining planning horizon. The default implementation is leaving it constant with the value of the last step of the control horizon
      MPCobj.ManipulatedVariables(1).Min (1:controlHorizon) = -maxAcc;
      MPCobj.ManipulatedVariables(1).Max (1:controlHorizon) = +maxAcc;
      MPCobj.ManipulatedVariables(2).Min (1:controlHorizon) = -maxAcc;
      MPCobj.ManipulatedVariables(2).Max (1:controlHorizon) = +maxAcc;
      MPCobj.ManipulatedVariables(3).Min (1:controlHorizon) = -maxAcc;
      MPCobj.ManipulatedVariables(3).Max (1:controlHorizon) = +maxAcc;
      MPCobj.ManipulatedVariables(1).Min (controlHorizon:planningHorizon) = 0;
      MPCobj.ManipulatedVariables(1).Max (
        controlHorizon:planningHorizon) = 0;
    MPCobj.ManipulatedVariables(2).Min (
        controlHorizon:planningHorizon) = 0;
    MPCobj.ManipulatedVariables(2).Max (
        controlHorizon:planningHorizon) = 0;
    MPCobj.ManipulatedVariables(3).Min (
        controlHorizon:planningHorizon) = 0;
    MPCobj.ManipulatedVariables(3).Max (
        controlHorizon:planningHorizon) = 0;


%       offlineTime_start=ceil(0.2*(2*pi/w)/tstep);
%       offlineTime_end=ceil(0.4*(2*pi/w)/tstep);
%         MPCobj.ManipulatedVariables(1).Min (1:
    offlineTime_start) = -maxAcc*0.1;
%         MPCobj.ManipulatedVariables(1).Max (1:
    offlineTime_start) = +maxAcc*0.1;
%         MPCobj.ManipulatedVariables(2).Min (1:
    offlineTime_start) = -maxAcc;
%         MPCobj.ManipulatedVariables(2).Max (1:
    offlineTime_start) = +maxAcc;
%         MPCobj.ManipulatedVariables(3).Min (1:
    offlineTime_start) = -maxAcc*0.1;
%         MPCobj.ManipulatedVariables(3).Max (1:
    offlineTime_start) = +maxAcc*0.1;
%         MPCobj.ManipulatedVariables(1).Min (
    offlineTime_start:offlineTime_end) = 0;
%         MPCobj.ManipulatedVariables(1).Max (
    offlineTime_start:offlineTime_end) = 0;
%         MPCobj.ManipulatedVariables(2).Min (
    offlineTime_start:offlineTime_end) = 0;
%         MPCobj.ManipulatedVariables(2).Max (
    offlineTime_start:offlineTime_end) = 0;
%         MPCobj.ManipulatedVariables(3).Min (
    offlineTime_start:offlineTime_end) = 0;
%         MPCobj.ManipulatedVariables(3).Max (
    offlineTime_start:offlineTime_end) = 0;
%         MPCobj.ManipulatedVariables(1).Min (
    offlineTime_end:planningHorizon) = -maxAcc*0.1;
%         MPCobj.ManipulatedVariables(1).Max (
    offlineTime_end:planningHorizon) = +maxAcc*0.1;
```

```
%       MPCobj.ManipulatedVariables(2).Min (
    offlineTime_end:planningHorizon) = −maxAcc;
%       MPCobj.ManipulatedVariables(2).Max (
    offlineTime_end:planningHorizon) = +maxAcc;
%       MPCobj.ManipulatedVariables(3).Min (
    offlineTime_end:planningHorizon) = −maxAcc∗0.1;
%       MPCobj.ManipulatedVariables(3).Max (
    offlineTime_end:planningHorizon) = +maxAcc∗0.1;




%% specify nominal values for inputs and outputs
MPCobj.Model.Nominal.U = [0;0;0];
MPCobj.Model.Nominal.Y = [0;0;0;0;0;0];
%% specify scale factors for inputs and outputs
MPCobj.MV(1).ScaleFactor = 0.0000002;    %min −100nN/m
              max +100nN/m  Can be +−8000nN and should be
     including timed thrust usage
MPCobj.MV(2).ScaleFactor = 0.0000002;
MPCobj.MV(3).ScaleFactor = 0.0000002;
MPCobj.OV(1).ScaleFactor = 4;             %min −2000m
             max +2000m      if x0=semimajor=2000
MPCobj.OV(2).ScaleFactor = 0.004;         %min −2.1236m/
    s    max +2.1236m/s   if dx0=x0∗angularSpeed
MPCobj.OV(3).ScaleFactor = 2;             %min −1000m
             max +1000m      if y0=init_y_chief_amplitude
    =1000
MPCobj.OV(4).ScaleFactor = 0.002;         %min −1.0618m/
    s    max +1.0618m/s  if dy0=y0∗angularSpeed or
    semimajor∗sqrt(0.75)∗omega  %∗0.001
MPCobj.OV(5).ScaleFactor = 2;             %min −1000m
             max +1000m      if z0=semimajor/2
MPCobj.OV(6).ScaleFactor = 0.002;         %min −1.0618m/
    s    max +1.0618m/s  if dz0=z0∗angularSpeed

%% use custom output disturbance model
%setoutdist(MPCobj, 'model', MPCobj_ModelOD);
%% use custom measurement noise model
%MPCobj.Model.Noise = MPCobj_ModelMN;

```

```
%% specify constraints for MV and MV Rate
% MPCobj.MV(1).Min = -0.5;
% MPCobj.MV(1).Max = 0.5;
%% specify weights
MPCobj.Weights.MV = [1 1 1];
 MPCobj.Weights.MVRate =0*[1 1 1]; %positive effect on unstable finish line effect , negative effect on spiked  efficient thrust behavior (positive in unefficient square behavior)
%  INPUTWeights = ones(planningHorizon, 3);          % Output weight  zero until end of planning horizon(finite Horizon LQR cost function)
%  INPUTWeights(planningHorizon,:) = [0 0 0]; % Output weight  zero until end of planning horizon(finite Horizon LQR cost function)
%  INPUTWeights(planningHorizon-1,:) = [0 0 0];
%   INPUTWeights(planningHorizon-2,:) = [0 0 0];
%    INPUTWeights(planningHorizon-3,:) = [0 0 0];
%     INPUTWeights(planningHorizon-4,:) = [0 0 0];
% MPCobj.Weights.MV = INPUTWeights;
%MPCobj.MV(1).Target=0.00002;
%MPCobj.MV(2).Target=-0.0002;
%MPCobj.MV(3).Target=0.00002;
outputWeights = zeros(planningHorizon, 6);          % Output weight  zero until end of planning horizon(finite Horizon LQR cost function)
outputWeights(planningHorizon,:) =1.7*[1 1 1 1 1 1]; % Output weight  zero until end of planning horizon(finite Horizon LQR cost function)
MPCobj.Weights.OV = outputWeights;                  % Output weight  zero until end of planning horizon(finite Horizon LQR cost function)
%MPCobj.Weights.OV = [1 1 1 1 1 1];

MPCobj.Weights.ECR = 1;  %Harder constraints, increased computation time, better results

MPCobj.Optimizer.MaxIter= 100;  %Limit the amount of time necessary to compute a solution
MPCobj.Optimizer.UseSuboptimalSolution=1; %Give the solution computed within the MaxIter limit
MPCobj.Optimizer.UseWarmStart=1;

setEstimator(MPCobj,'custom');%necessary for simulink

% [coredata,statedata,onlinedata] =
    getCodeGenerationData(MPCobj);
% fun = 'mpcmoveCodeGeneration';
% funOutput = 'mpcmoveMEX';
% Cfg = coder.config('mex');
% Cfg.DynamicMemoryAllocation = 'off';
% codegen('-config',Cfg,fun,'-o',funOutput,'-args',...
%     {coder.Constant(coredata),statedata,onlinedata})
    ;
```

Dynamic model

```
%Panayiotis D. Kremmydas
%Find initial values that when used in the CW
    equations, the resulting
%plots ressemble particular circular or eleptical
    motions in the LO axes.

%%RADS: radians/second

function [x,y,z,dx,dy,dz] = FreeDriftFormations(x0,y0,
    z0,dx0,dy0,dz0,omeg,t)
x    = ((4*dx0/omeg) - 6*z0)*sin(omeg*t)  -  (2*dz0/
    omeg)*cos(omeg*t)  +  (6*omeg*z0-3*dx0)*t  +  (x0
    +(2*dz0/omeg));
y    = y0*cos(omeg*t) +  (dy0/omeg)*sin(omeg*t);
z    = ((2*dx0/omeg) - 3*z0)*cos(omeg*t) + (dz0/omeg)*
    sin(omeg*t) + (4*z0- (2*dx0/omeg));

dx  = (4*dx0-6*z0*omeg)*cos(omeg*t)+2*dz0*sin(omeg*t)
    +6*z0*omeg-3*dx0;
dy  = -y0*omeg*sin(omeg*t)+dy0*cos(omeg*t);
dz  = (3*z0*omeg-2*dx0)*sin(omeg*t)+dz0*cos(omeg*t);
return
%%%could be much lighter computationally if for a
    computation iterration
```

```
%%%interval of 10 samples per second, adding to the location vector the
%%%(velocity vector mean for that interval) * (interval time)


%%Digital way of robot(no acceleration): Taking a location and a given speed
%%(Calculate new optimal speed) Finding the next location according to the x,y,z+= (new_velocity)*(time) interval between that location and
%%the next.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Iteration based calculation: Acceleration is not time dependant, all
%%other quantities can be calculated based on the previous step velocity
%%and position (which gives the approximate acceleration for the current interval),
%%and approximated given the iteration time  xdot+= xdotdot * t , x+=xdot*t
```

Moving Plane formation generator

```
function  [x,y,z,dx,dy,dz]= cw_moving_plane(init_y_amplitude,semi_major_xz_plane_ellipse,satellite_number,angularSpeed,t)
%%%%%CW Moving Plane Formation (Moving Y bar on YZ and YX axes): x0=(+-k)*arbitrary, y0=arbitrary , z0=0, dx0=0 , dy0=arbitrary , dz0=(+-k)*(0.5)*omega*x0,  where k is the corresponding number of the satellite
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\\\\\\ SatelliteFormation  NORMAL LVLH(X,Y,Z)
%semi_major_xz_plane_ellipse=400; % coincides with the  x-axis initial distance of the required orbit in meters, semi-minor on z-axis is always half (orbital mechanics) %asuming that the initial z0 is
```

```
set to 0 for the first of the satellites
dz0_ellipse= -(semi_major_xz_plane_ellipse*angularSpeed)/2; %velocity necessary to form an ellipse centered on x=0,z=0
%init_y_amplitude=800; %; % because starting point is x0=semi_major_xz_plane_ellipse an initial y0 amplitude will give a yaw to the formation elipse
init_y_velocity=1.234567*dz0_ellipse;% because starting point is x0=semi_major_xz_plane_ellipse an initial dy0 velocity will give a roll to the formation elipse
%satellite_number=4;

%When y(t) is at its peak when z(t) is at its peak (t=pi/2 for the conditions above) and y(pi/2)=sqrt(0.75)*semi_major_xz_plane_ellipse, then the satellites are orbiting in constant distance to the origin
%y(t)+z(t)=x(t) when ymax(t)^2+zmax(t)^2=xmax(t)^2, zmax and xmax linked with semimazor axis
%init_y_velocity=sqrt(0.75)*semi_major_xz_plane_ellipse*angularSpeed=866.0254*0.00113027258
s=length(t);
for k = 1:satellite_number
    if mod(k,2);
     [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k,1:s),dz(k,1:s)] = FreeDriftFormations(k*semi_major_xz_plane_ellipse,init_y_amplitude,0,0,init_y_velocity,k*dz0_ellipse,angularSpeed,t);
    else
     [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k,1:s),dz(k,1:s)] = FreeDriftFormations(-(k-1)*semi_major_xz_plane_ellipse,init_y_amplitude,0,0,init_y_velocity,-(k-1)*dz0_ellipse,angularSpeed,t);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Projected Circular Orbit formation generator

```

        function  [x,y,z,dx,dy,dz]= cw_pco(semi_major_xz_plane_ellipse , satellite_number , angularSpeed,t)
          T=2*pi/angularSpeed;
          %%%%%Projected Circular Orbit (on X-Y plane): x0=arbitrary, y0=arbitrary , z0=0 , dx0=0 , dy0=-2dz0 , dz0=(0.5)*omega*x0
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\\\\\\\ SatelliteFormation  NORMAL LVLH(X,Y,Z)
          %%%the following 3 variables perfectly define every possible single plane force free formation: x0, z0 , dx0 , dz0 are always dependant for a FFF centered at origin, one defines the others)
          %semi_major_xz_plane_ellipse=1000; % coincides with the x-axis initial distance of the required orbit in meters, semi-minor on z-axis is always half (orbital mechanics) %asuming that the initial z0 is set to 0 for the first of the satellites
          %satellite_number=4;
          dz0_ellipse= -(semi_major_xz_plane_ellipse*angularSpeed)/2; %velocity necessary to form an ellipse centered on x=0,z=0
          init_y_amplitude=0; %; % because starting point is x0=semi_major_xz_plane_ellipse an initial y0 amplitude will give a yaw to the formation elipse
          init_y_velocity=2*dz0_ellipse;% because starting point is x0=semi_major_xz_plane_ellipse an initial dy0 velocity will give a roll to the
          formation ellipse

        %When y(t) is at its peak when z(t) is at
          its peak (t=pi/2 for the conditions
          above) and y(pi/2)=sqrt(0.75)*
          semi_major_xz_plane_ellipse, then the
          satellites are orbiting in constant
          distance to the origin
        %y(t)+z(t)=x(t) when ymax(t)^2+zmax(t)^2=
          xmax(t)^2,  zmax and xmax linked with
          semimazor axis
        %init_y_velocity=sqrt(0.75)*
          semi_major_xz_plane_ellipse*
          angularSpeed=
          866.0254*0.00113027258
        s=length(t);
        for k = 1:satellite_number
            [xt,yt,zt,dxt,dyt,dzt] =
              FreeDriftFormations(
              semi_major_xz_plane_ellipse,
              init_y_amplitude,0,0,init_y_velocity
              ,dz0_ellipse,angularSpeed,((k-1)/
              satellite_number)*T);
            [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),
              dy(k,1:s),dz(k,1:s)] =
              FreeDriftFormations(xt,yt,zt,dxt,dyt
              ,dzt,angularSpeed,t);
        end
    return
```

3 in plane, 1 out of plan formation generator

```
    function  [x,y,z,dx,dy,dz]= cw_plane_1out(
        init_y_chief_amplitude,
        semi_major_xz_plane_ellipse,
        satellite_number,angularSpeed,t)
    T=2*pi/angularSpeed;
    %%%%%CW one plane formation , 1 out of plane:
        x0=arbitrary, y0=arbitrary , z0=0 , dx0=0 ,
         dy0=-2dz0 , dz0=(0.5)*omega*x0
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\\\\\\\ SatelliteFormation  NORMAL LVLH(X,Y,Z)
%%%the following 3 variables perfectly define every possible single plane force free formation: x0, z0 , dx0 , dz0 are always dependant for a FFF centered at origin, one defines the others)
%semi_major_xz_plane_ellipse=1000; % coincides with the x-axis initial distance of the required orbit in meters, semi-minor on z-axis is always half (orbital mechanics) % asuming that the initial z0 is set to 0 for the first of the satellites
 dz0_ellipse= -(semi_major_xz_plane_ellipse*angularSpeed)/2; %velocity necessary to form an ellipse centered on x=0,z=0
 init_y_deputy_amplitude=0; %; % because starting point is x0=semi_major_xz_plane_ellipse an initial y0 amplitude will give a yaw to the formation elipse
 init_y_deputy_velocity=0;% because starting point is x0=semi_major_xz_plane_ellipse an initial dy0 velocity will give a roll to the formation elipse
%init_y_chief_amplitude=100;
 init_y_chief_velocity=0.8;
%satellite_number=4;

%When y(t) is at its peak when z(t) is at its peak (t=pi/2 for the conditions above) and y(pi/2)=sqrt(0.75)*semi_major_xz_plane_ellipse, then the satellites are orbiting in constant distance to the origin
%y(t)+z(t)=x(t) when ymax(t)^2+zmax(t)^2=xmax(t)^2,  zmax and xmax linked with semimazor axis
%init_y_velocity=sqrt(0.75)*semi_major_xz_plane_ellipse*angularSpeed=
        866.0254*0.00113027258
        s=length(t);
        [x(1,1:s),y(1,1:s),z(1,1:s),dx(1,1:s),dy(1,1:s),dz(1,1:s)] = FreeDriftFormations(0,init_y_chief_amplitude,0,0,init_y_chief_velocity,0,angularSpeed,t);
        for k = 2:satellite_number
            [xt,yt,zt,dxt,dyt,dzt] = FreeDriftFormations(semi_major_xz_plane_ellipse,init_y_deputy_amplitude,0,0,init_y_deputy_velocity,dz0_ellipse,angularSpeed,((k-1)/(satellite_number -1))*T);
            [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k,1:s),dz(k,1:s)] = FreeDriftFormations(xt,yt,zt,dxt,dyt,dzt,angularSpeed,t);
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Equidistant formation generator

```
%Panayiotis D. Kremmydas
function  [x,y,z,dx,dy,dz]= cw_equidistant(semi_major_xz_plane_ellipse,satellite_number,angularSpeed,t)
%%%%%CW Equidistant Formation: x0=arbitrary, y0=0 , z0=0 , dx0=0 , dy0=x0*sqrt(0.75)*omega , dz0=x0*(0.5)*omega
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\\\\\\\ SatelliteFormation  NORMAL LVLH(X,Y,Z)0
%%the following 3 variables perfectly define every possible single plane force free formation: x0, z0, dx0 , dz0 are always dependant for a FFF centered at origin, one defines the others)
%semi_major_xz_plane_ellipse=1000; % coincides with the x-axis initial distance of the required orbit in meters, semi-minor on z-axis is always half (orbital mechanics) %asuming that the initial z0 is
```

```
set to 0 for the first of the satellites
dz0_ellipse= -(semi_major_xz_plane_ellipse*0.5*angularSpeed); %velocity necessary to form an ellipse centered on x=0,z=0
init_y_amplitude=0; %; % because starting point is x0=semi_major_xz_plane_ellipse an initial y0 amplitude will give a yaw to the formation elipse
init_y_velocity=sqrt(0.75)*semi_major_xz_plane_ellipse*angularSpeed;% because starting point is x0=semi_major_xz_plane_ellipse an initial dy0 velocity will give a roll to the formation elipse
T=2*pi/angularSpeed;
%satellite_number=4;
%When y(t) is at its peak when z(t) is at its peak (t=pi/2 for the conditions above) and y(pi/2)=sqrt(0.75)*semi_major_xz_plane_ellipse, then the satellites are orbiting in constant distance to the origin
%y(t)+z(t)=x(t) when ymax(t)^2+zmax(t)^2=xmax(t)^2, zmax and xmax linked with semimazor axis
%init_y_velocity=sqrt(0.75)*semi_major_xz_plane_ellipse*angularSpeed=866.0254*0.00113027258
s=length(t);
for k = 1:satellite_number
    [xt,yt,zt,dxt,dyt,dzt] = FreeDriftFormations(semi_major_xz_plane_ellipse,init_y_amplitude,0,0,init_y_velocity,dz0_ellipse,angularSpeed,((k-1)/satellite_number)*T);
    [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k,1:s),dz(k,1:s)] = FreeDriftFormations(xt,yt,zt,dxt,dyt,dzt,angularSpeed,t);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Tetrahedral formation generator

```
function [x,y,z,dx,dy,dz]= tetrahedral(
    semi_major_xz_plane_ellipse,inclination,
    angularSpeed,t)
%%%%%Tetrahedral Formation (constant volume inside
    formation): chief: x0=arbitrary, y0=x0/2 , z0=0 ,
    dx0=0 , dy0=0 , dz0=x0*(0.5)*omega
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\\\\\\\
    SatelliteFormation  NORMAL LVLH(X,Y,Z)
s=length(t);
lenght_of_tetrahedron_x_axis=2*
    semi_major_xz_plane_ellipse;
A=0.3*(sqrt(3)/4)*lenght_of_tetrahedron_x_axis;
%inclination=+1;%1 or -1, y and dy can also be
    inversed to achieve inversed inclination of the
    local orbit of the 2 satellites

[x(1,1:s),y(1,1:s),z(1,1:s),dx(1,1:s),dy(1,1:s),dz
    (1,1:s)] = FreeDriftFormations(-A*(2/3),-
    inclination*A*sqrt(3),2*sqrt(2)*A/3,+2*angularSpeed
    *2*sqrt(2)*A/3,0,+angularSpeed*A/3,angularSpeed,t);
[x(2,1:s),y(2,1:s),z(2,1:s),dx(2,1:s),dy(2,1:s),dz
    (2,1:s)] = FreeDriftFormations(2*A,A/sqrt(3),0,0,
    inclination*2*sqrt(2)*A*angularSpeed/sqrt(3),-
    angularSpeed*A,angularSpeed,t);
[x(3,1:s),y(3,1:s),z(3,1:s),dx(3,1:s),dy(3,1:s),dz
    (3,1:s)] = FreeDriftFormations(-
    lenght_of_tetrahedron_x_axis/2,0,0,0,0,0,
    angularSpeed,t);
[x(4,1:s),y(4,1:s),z(4,1:s),dx(4,1:s),dy(4,1:s),dz
    (4,1:s)] = FreeDriftFormations(
    lenght_of_tetrahedron_x_axis/2,0,0,0,0,0,
    angularSpeed,t);

%semi_major_xz_plane_ellipse=
    lenght_of_tetrahedron_x_axis/3;
%satellite_number=4;
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Helix formation generator

```
function  [x,y,z,dx,dy,dz]= cw_helix(semi_major_xz_plane_ellipse,satellite_number,angularSpeed,t)
T=2*pi/angularSpeed;
%%%%%CW Helix Formation: chief: x0=arbitrary , y0=x0/2 , z0=0 , dx0=0 , dy0=0 , dz0=x0*(0.5)*omega
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\\\\\\ SatelliteFormation NORMAL LVLH(X,Y,Z)
%semi_major_xz_plane_ellipse=1000; % coincides with the x-axis initial distance of the required orbit in meters, semi-minor on z-axis is always half (orbital mechanics) %asuming that the initial z0 is set to 0 for the first of the satellites
dz0_ellipse= -(semi_major_xz_plane_ellipse*0.5*angularSpeed); %velocity necessary to form an ellipse centered on x=0,z=0
init_y_amplitude=semi_major_xz_plane_ellipse/2; %; % because starting point is x0=semi_major_xz_plane_ellipse an initial y0 amplitude will give a yaw to the formation elipse
init_y_velocity=0;% because starting point is x0=semi_major_xz_plane_ellipse an initial dy0 velocity will give a roll to the formation elipse
%satellite_number=4;
%When y(t) is at its peak when z(t) is at its peak (t=pi/2 for the conditions above) and y(pi/2)=sqrt(0.75)*semi_major_xz_plane_ellipse, then the satellites are orbiting in constant distance to the origin
%y(t)+z(t)=x(t) when ymax(t)^2+zmax(t)^2=xmax(t)^2, zmax and xmax linked with semimazor axis
%init_y_velocity=sqrt(0.75)*semi_major_xz_plane_ellipse*angularSpeed=866.0254*0.00113027258
s=length(t);
for k = 1:satellite_number
    [xt,yt,zt,dxt,dyt,dzt] = FreeDriftFormations(semi_major_xz_plane_ellipse,init_y_amplitude
        ,0,0,init_y_velocity,dz0_ellipse,angularSpeed,((
        k-1)/satellite_number)*T);
    [x(k,1:s),y(k,1:s),z(k,1:s),dx(k,1:s),dy(k,1:s),dz(
        k,1:s)] = FreeDriftFormations(xt,yt,zt,dxt,dyt,
        dzt,angularSpeed,t);
end
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Alternative main function for perpetual control

```
%% 2: Formation CONTROL Acquisition Transfer
    if      formation2==1          %cw_equidistant
                [x1,y1,z1,dx1,dy1,dz1]=   ...
            cw_equidistant(semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,0);
    elseif  formation2==2          %cw_pco
                [x1,y1,z1,dx1,dy1,dz1]=   ...
            cw_pco(semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,0);
    elseif  formation2==3          %cw_pco
                [x1,y1,z1,dx1,dy1,dz1]=   ...
            cw_plane_1out(init_y_chief_amplitude,
                semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,0);
    elseif  formation2==4          %cw_pco
                [x1,y1,z1,dx1,dy1,dz1]=   ...
            cw_equidistant(semi_major_xz_plane_ellipse,
                satellite_number,angularSpeed,0);
    elseif  formation2==5          %cw_pco
                [x1,y1,z1,dx1,dy1,dz1]=   ...
            tetrahedral(semi_major_xz_plane_ellipse,
                inclination,angularSpeed,t_freedrift);
    end

    offline_steps=ceil((2*pi/angularSpeed)/
        sampleInterval)

    for k = 1:satellite_number
```

```
            %FreeDriftFormations(x0,y0,z0,dx0,dy0,dz0,
               omeg,t) values propagate according to
               the CW equations for formation1 orbits
            s0 = [xtot(k,t1_len);  dxtot(k,t1_len);
               ytot(k,t1_len); dytot(k,t1_len);  ztot(k
               ,t1_len); dztot(k,t1_len)]; %initial
               spacecraft state before control
               initiation
            sr = [x1(k); dx1(k); y1(k); dy1(k); z1(k);
               dz1(k)]; %state that belongs to the
               defined second formation to be acquired:
                formation2
            y= s0;  %initial spacecraft state before
               control initiation

            %control routine:
             for i=2:offline_steps:t2_len+offline_steps
                    [u] = ctrl_mpc_transfer_hill(s0
                       (1), ...
                                                  s0(3),
                                                  ...
                                                  s0(5),
                                                  ...
                                                  s0(2),
                                                  ...
                                                  s0(4),
                                                  ...
                                                  s0(6),
                                                  ...
                                                  sr(1),
                                                  ...
                                                  sr(3),
                                                  ...
                                                  sr(5),
                                                  ...
                                                  sr(2)
                                                  ,...
                                                  sr(4)
                                                  ,...
                                                  sr(6)
                                                  ,...
                                                  angularSpeed
                                                  ,
                                                  sampleInterval
                                                  );

                      %trajectory simulated states
                        using the given multistep
                       thrust vector {normally
                       replaced by orkit advanced
                       simulation}
                    for j=0:offline_steps-1
                        if j< ceil((2/3)*(2*pi/
                          angularSpeed)/
                          sampleInterval) %
                          before eclipse {lsim
                          with thrust
                          utilization}
                            y_move = lsim(
                              statespace,[u(3*j
                              +1:3*j+3) ; zeros
                              (1,3)],0:
                              sampleInterval:
                              sampleInterval,y
                              (:,i+j-1));
                            y(:,i+j) = y_move(2,:)
                              ;
                        else %after eclipse {lsim
                          without thrust
                          utilization}
                            y_move = lsim(
                              statespace ,[ zeros
                              (1 ,3)  ;  zeros(1 ,3)
                              ] ,0: sampleInterval :
                              sampleInterval ,y(: ,
                              i+j −1)) ;
                          y(: , i+j )  =  y_move(2 ,:)
                              ;
                      end
                 end
                    s0  =  y_move(2 ,:) ;   %final
                       simulated  state  becomes  the
                       target  state  for  the  next
                       control  iteration
                    [rtemp ,~ ,~]  =  lsim(statespace ,
                       zeros(3 ,offline_steps+1) ,  0:
                       sampleInterval : sampleInterval
                       ∗(offline_steps) ,sr) ;
                    sr=rtemp(offline_steps+1 ,:) ’; %
                       propagated  target  state :
                       should  include  the  number  of
                       steps  that  the  current
                       satellite  state  has  been
                       propagated  since  last  mpc  call
                     %(e.g.  steps  propagated  using
                        the  MPC  inputs  or  steps
                        propagated  using  zero  inputs
                        in  the  case  of  the  eclipse)
              end
               y=y(: ,2: t2_len+1) ; %generated  trajectory
                   for  the  overall  simulation
               xtot(k, t1_len+1  :  t1_len+t2_len)=y(1 ,:) ;
               ytot(k, t1_len+1  :  t1_len+t2_len)=y(3 ,:) ;
               ztot(k, t1_len+1  :  t1_len+t2_len)=y(5 ,:) ;
              dxtot(k, t1_len+1  :  t1_len+t2_len)=y(2 ,:) ;
              dytot(k, t1_len+1  :  t1_len+t2_len)=y(4 ,:) ;
              dztot(k, t1_len+1  :  t1_len+t2_len)=y(6 ,:) ;
              sum(sum(abs(((100∗u)∗(100∗u’))∗
                 sampleInterval)))
              sum(sum(abs(u∗sampleInterval)))
          %%%%%%%%%%%%%%%%%
          display(’satellite
              :///////////////////////////////////////////////////////////////
              ', num2str(k))
     end
             %%%%%%%%%%%%%%%%%%%
```

Alternative MPC wrapper for perpetual control

```
function [ u ] = ctrl_mpc_transfer_hill( x0,y0,z0, dx0
    ,dy0,dz0,x1,y1,z1,dx1,dy1,dz1, angularSpeed,
    sampleInterval) %w=angular frequency m %tstep=
    sample time
%CTRL_HILL MPC based on
%Co-ordination and control of distributed spacecraft
%systems using convex optimization techniques
%   Detailed explanation goes here
    %start mean [a theta i q1 q2 Omega lambda e]
    %final time (offset from now) Tf [s]
    % target vector at Tf: Xfinal
    %time step Ts (integration step size)


    %%%%%%%First run, mpc initialization.
    persistent VarName;
    persistent MPCobj;
    persistent options;
    persistent x;
    if isempty(VarName) %Can cause problems when
        changing angularSpeed in the calling function
        [MPCobj]= netSATmpc(angularSpeed,
            sampleInterval);
        x = mpcstate(MPCobj);  %Current state
        options = mpcmoveopt; %Used for possible
            weight allocation on runtime
        VarName = 0;
    end
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


      %disp('    >>>> MPC CTRL TRANSFER HILL');
```

```
        s0 = [x0; dx0; y0; dy0; z0; dz0]; %Current State
        sr = [x1; dx1; y1; dy1; z1; dz1]; %Propagated Reference State (according to the current timestep of the simulation)
        x.Plant= s0;            %internal state of the mpc controller needs to be initiallized.
       [unused,mpcmoveInfo]= mpcmove(MPCobj, x, x.Plant, sr   , [], options);
     for k=0:1:size(mpcmoveInfo.Uopt)-2
         u(3*k+1:3*k+3)=mpcmoveInfo.Uopt(k+1,:);
     end
    % u
    % pause


      %% Dispaying inputs
%       disp('X_0_LVLV:');
%       disp(X_0_LVLV);
%       disp('X_T_LVLH:');
%       disp(X_T_LVLH);
%       disp('orbitSteps:');
%       disp(orbitSteps);
%       disp('controlHorizon:');
%       disp(controlHorizon);
%       disp('stepDuration:');
%       disp(stepDuration);


      %% Return value u
      % If computation successful, return an (1 x n) matrix consisting of
      % concatenated control vectors (acceleration in m/s).
      % n = 3 * ceil(controlHorizon_s / steps)
end
```

Simulation animation file generator

```
%Panayiotis D. Kremmydas
function inLoopTakeFrame_AppendToGif(k)
```

```
    im = frame2im(getframe);
    [imind,cm] = rgb2ind(im,256);
    % Write to the GIF File
    if k == 1
        imwrite(imind,cm,'Animated.gif','gif', '
          Loopcount',inf,'DelayTime', 1/60);
    else
        imwrite(imind,cm,'Animated.gif','gif','
          WriteMode','append','DelayTime', 1/60);
    end
return
```

Bibliography

[BBC12] S. K. Barik, M. P. Biswal, and D. Chakravarty. Multiobjective two-stage stochastic programming problems with interval discrete random variables. *Advances in Operations Research*, 2012:1–21, 2012.

[BH08] Louis S. Breger and Jonathan P. How. Safe Trajectories for Autonomous Rendezvous of Spacecraft. *Journal of Guidance, Control, and Dynamics*, 31(5):1478–1489, 2008.

[CW60] R S Clohessy and Wiltshire. The Clohessy-Wiltshire Equations of Relative Motion. *Journal of Aerospace Sciences*, 27(9):653–658, 1960.

[Feh03] W. Fehse. Automated Rendezvous and Docking of Spacecraft. *Automated Rendezvous and Docking of Spacecraft*, pages 98–106, 2003.

[Hil78] G.W. Hill. Researches in the Lunar Theory. *American Journal of Mathematics*, 1(3):245–260, 1878.

[NM13] Austin K. Nicholas and David W. Miller. Attitude and formation control design and system simulation for a three-satellite CubeSat mission. (June), 2013.

[PSK15] Vryan Gil Palma, Andrea Suardi, and Eric C. Kerrigan. Sensitivity-based multistep MPC for embedded systems. *IFAC-PapersOnLine*, 48(23):360–365, 2015.

[SCSP08] Space Station, Reference Coordinate, International Space, and Station Program. Esa. 2008.

[SS02] Samuel A. Schweighart and Raymond J. Sedwick. High-Fidelity Linearized J2 Model for Satellite Formation Flight. *Journal of Guidance, Control, and Dynamics*, 25(6):1073–1080, 2002.