FLOWER CONSTELLATION OPTIMIZATION AND IMPLEMENTATION

A Dissertation

by

CHRISTIAN BRUCCOLERI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2007

Major Subject: Aerospace Engineering

FLOWER CONSTELLATION OPTIMIZATION AND IMPLEMENTATION

A Dissertation

by

CHRISTIAN BRUCCOLERI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Daniele Mortari
Committee Members,	John L. Junkins
	Thomas C. Pollock
	J. Maurice Rojas
Head of Department,	Helen Reed

December 2007

Major Subject: Aerospace Engineering

ABSTRACT

Flower Constellation Optimization and Implementation. (December 2007)

Christian Bruccoleri, M.S., Universitá di Roma - La Sapienza

Chair of Advisory Committee: Dr. Daniele Mortari

Satellite constellations provide the infrastructure to implement some of the most important global services of our times both in civilian and military applications, ranging from telecommunications to global positioning, and to observation systems. Flower Constellations constitute a set of satellite constellations characterized by periodic dynamics. They have been introduced while trying to augment the existing design methodologies for satellite constellations. The dynamics of a Flower Constellation identify a set of implicit rotating reference frames on which the satellites follow the same closed-loop relative trajectory. In particular, when one of these rotating reference frames is "Planet Centered, Planet Fixed", then all the orbits become compatible (or resonant) with the planet; consequently, the projection of the relative path on the planet results in a repeating ground track.

The satellite constellations design methodology currently most utilized is the Walker Delta Pattern or, more generally, Walker Constellations. The set of orbital planes and initial spacecraft positions are represented by a set of only three integers and two real parameters rather than by all the orbital elements; Flower Constellations provide a more general framework in which most of the former restrictions are removed, by allowing the use of resonant elliptical orbits. Flower Constellations can represent hundreds of spacecraft with a set of 6 integers and 5 real parameters only and existing constellations can be easily reproduced.

How to design a Flower Constellation to satisfy specific mission requirements

is an important problem for promoting the acceptance of this novel concept by the space community. Therefore one of the main goals of this work is that of proposing design techniques that can be applied to satisfy practical mission requirements.

The results obtained by applying Global optimization techniques, such as Genetic Algorithms, to some relevant navigation and Earth observation space-based systems show that the Flower Constellations not only are as effective as Walker Constellations, but can also be applied to non-traditional constellation problem domains, such as regional coverage and reconnaissance.

"*Fatti non foste a viver come bruti*
ma per seguire virtute e canoscenza"

Dante Alighieri, *Divina Commedia*,
Inferno canto XXVI, 116-120

to Francesco and Orietta

ACKNOWLEDGMENTS

Accomplishing anything non trivial in life is a long, and often hard, journey. One may only thank his good luck if such a journey is blessed by the help and advice of friends and scholars who ventured upon similar paths before. In the years that I spent trying to achieve this important goal in my life, many people sustained this effort in different ways. I believe that gratitude to my advisor and friend, Dr. Daniele Mortari, can only be expressed with a simple "*Thanks!*", for I know that he likes simple things and short speeches, whereas explaining all the ways in which he, together with Andreea and Anna, helped me would take a chapter of its own.

Similarly, I know that Dr. John L. Junkins asks no more satisfaction from his students than seeing them going ahead on their own and accomplishing things in life, both professionally and personally. I must therefore thank him for his precious advice, but above all for his unshaken optimism and foresight. To Dr. Thomas C. Pollock I wish to express my gratitude for his technical advice and for the friendly and comfortable way in which he always approaches his students.

I would have had to stop my journey much earlier, however, if it were not for the providential help of StarVision Technologies, Inc. which sponsored my Ph.D. for more than three years, in spite of the difficulties that this proposition entailed. I express my deepest gratitude to Mr. Michael Jacox and Dr. James A. Ochoa for their fundamental support and advice during times in which they were heavily involved in the making of a successful private company. Similarly, I wish to thank all the StarVision Technologies employees for their friendliness. I wish to thank Dr. Walter C. Haisler and Karen Knabe for their responsiveness to the many administrative issues that students face during their career; they were always ready to help and point me in the right direction.

Many others have accompanied me on various parts of this journey; I wish I could mention them all, but it would not only grow tedious to the reader; it would also make too many words of thanks fade in meaning and intensity. So, I will simply mention some of my friends and colleagues who made a difference in my daily life like Randy and Lori South, Dr. Giovanni Giardini, Dr. Anup Katake, Dr. Puneet Singla, Dr. Matthew P. Wilkins, Troy Henderson, Marco De Santis, Alberto Clocchiatti, and this list could go on for much longer. Finally, I shall remind myself, and the unlikely readers, that every accomplishment is just a new beginning and the climb starts anew: ever towards more misty peaks.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

This dissertation is aimed at addressing applications and relative design issues of Flower Constellations by developing re-usable application software and providing significative examples in some of the main areas in which satellite constellations are being used or will be used in the future.

Artificial satellite constellations constitute a fundamental infrastructure for delivering some of the most widely used and mission-critical global services of our times. The importance of real-time telecommunications, global navigation systems, Earth monitoring for disaster prevention or recovery, Earth science, climate observation, and space-based surveillance hardly needs to be stressed at all. The explosion in number and capabilities of small portable electronic devices, coupled with a dramatic acceleration in the economies of Asian countries like China and India, make us witness nowadays an unprecedented growth in the demand of even more advanced and reliable services in the areas just mentioned above.

On the other hand, however, the space-technology for supporting such advanced services is somewhat lagging behind. Electronic devices validated for use in outer space are at least ten years behind comparable COTS[1] electronics for use on Earth and the cost of sending satellites in orbit is still very high: so much that only governments and the biggest corporations can afford it.

One way to achieve cost reduction is the development of smaller, redundant satellites that act cooperatively to achieve mission goals, i.e. micro and nano satellites

---

The journal model is *IEEE Transactions on Automatic Control.*

[1] Commercial Off The Shelf

operating in formations[2]. Designing such smaller crafts presents many challenges of its own, but it is a well established trend of the last ten years, in response to the problems of a) sharing the burden of launch costs and risks between several entities, and b) providing a more robust architecture through redundancy.

Working cooperatively in space can be achieved with two similar multi-spacecraft schemes: formation flying and constellations. The difference in the definition is somewhat blurred, but in general it is accepted within the space community that formation flying involves spacecrafts orbiting from few meters to few kilometers from each other, and not exceeding some predefined maximum distance, small compared to an Earth radius, whereas constellations can have satellites participating to the mission goals in different orbits and thousands of kilometers apart. Another way of differentiating the concepts is that in formation flying spacecrafts are always in view of each other, i.e. no spacecraft is eclipsed by Earth, in constellations this is not necessarily required.

This dissertation presents the development and testing of design methodologies for Flower Constellations, a recently introduced framework for constellation design that encompasses existing constellations and that can generate entirely new types of configurations. Applications that benefit from innovative constellation design are considered: mission profiles for global navigation, telecommunication, and regional coverage are examined and compared with existing or proposed constellations. How to design symmetric Flower Constellations for global navigation and how to design configurations for regional coverage is presented in the following chapters, together with the development of the necessary theoretical and software tools, such as the FCVAT program, the fast orbital propagation routines, and the FCToolbox for MATLAB. These tools will be described in detail within the implementation section.

[2]Micro satellites are satellites that weigh from 10 up to 100 Kg and nano satellites weigh up to 10 Kg.

The most representative and well known constellation actually in orbit at the time of this writing is probably the USNO NAVSTAR Global Positioning System, more widely known simply as GPS [1, 2, 3, 4]. The GPS constellation consists of 24 satellites in six orbital planes, with four operational satellites in each plane, plus spares. At this time there are actually 30 satellites in the constellation since some older spacecrafts are going to be decommissioned and there are spare satellites to maintain level of service in case of unexpected failure of any of the operational satellites. The satellites operate in circular orbits at an altitude of $20,200$ km, inclination angle of 55 degrees, for a 12 hour orbital period. The constellation is designed so that the position of the satellites is the same at the same sidereal time[3] each day.

A similar constellation, in purpose and layout, is GalileoSat [5], which is going to be completed under the supervision of the European Union. Galileo will have 30 satellites in Medium Earth Orbit (MEO) at an altitude of 23,222 km. Ten satellites will occupy each of three orbital planes inclined at an angle of 56 degrees. The satellites will be uniformly distributed around each plane and will have a 14 hour orbital period. One satellite in each plane will be a spare, on stand-by should any operational satellite fail.

Another similar system is GLONASS, developed by USSR and partially deactivated with the economical and political problems during the years 1990s in that country. A restoration of the system by Russia, with India as important partner, is scheduled for 2011. A characteristic of the GLONASS constellation is that any given satellite passes over the exact same point on the Earth surface every eighth sidereal days. This characteristic is also known as having a *repeating ground track*, a property that is also utilized for Flower Constellations as will be explained in the next chapter.

[3]A definition of sidereal time will be provided in the next chapter

GalileoSat and GLONASS can be described as Walker Delta Patterns constellations [6, 7]. All navigation systems share the same goal: seeing from any point on the Earth surface a sufficient number of spacecrafts to achieve the desired level of service in position accuracy. It is therefore important that satellites are uniformly distributed, remaining always well spread to provide a good Geometrical Dilution of Precision (GDOP) and ultimately accurate position and velocity measurements. It will be shown in the following that Walker Delta patterns can be reproduced within the Flower Constellation framework and, therefore, the latter methodology includes existing solutions.

Many communication satellites are placed in *Geostationary* orbit, in which a spacecraft orbits the Earth with a period of 23 hours and 56 minutes (one *sidereal day*) at an altitude of $35,790$ Km, thus matching the Earth rotation period. Satellites in Geostationary, or Geo, orbits linger always above a determined region and, therefore, are ideal for communication purposes. A large number of spacecrafts is nowadays occupying the Geo belt causing issues with assignment of frequencies and orbital slots. Beside these problems, a Geo satellite is generally much bigger, and thus expensive, than lower Earth orbit satellites: it needs a more expensive propulsion system to achieve the operational orbit, bigger solar arrays for increased power, and resilience to a higher dose of radiation during the extended life span. For these reasons, redundancy of key components for increased durability and reliability is essential: a fact that drives the cost of building and deploying a Geo satellite even higher.

A Geosynchronous constellation for telecommunication purposes, such as TDRS, covers almost the whole Earth surface with only 3 spacecrafts as shown in [8]. A drawback of Geo orbits is that while they cover the equatorial areas very well, they offer much poorer performances at the northern or southern latitudes. Another example of Geosynchronous constellation, for weather forecasts is GOES, with 4 satellites op-

erated by NOAA for the National Weather Service. Geostationary orbits can be seen as particular Flower Constellations as well.

While Constellations for global navigation are symmetric and uniformly distributed around the Earth, when Earth Observation and Space Surveillance mission are needed the coverage requirements are often limited to the continuous coverage above a specific region of interest. Thus, when regional coverage is desired, asymmetric constellations provide the best performance. The ultimate regional coverage can be provided by geostationary satellites, with the advantages and the problems described above. A serious concern, for surveillance missions or disaster monitoring, is the time required to deploy an operational satellite network after the materialization of a new threat or disaster: this almost certainly rules out geostationary systems unless already in place. The U.S. Air Force developed the concept of *Responsive Space* which defines the architecture and requirements of the ground and space segments for the quick assembly and deployment of new satellites, or the re-deployment of existing assets, in response to various kind of threats. This program fosters the utilization of expendable small spacecrafts with reduced life time expectancy and lower cost. Such spacecrafts should be assembled and deployed rapidly through *plug and play* components, within hours or at most days from the emergency situation. A relatively cheaper way to obtain the benefits of geostationary orbits is the use of the so called Molniya orbits: highly elliptical orbits in which the spacecrafts spend most of the orbiting time at the apogee that is located close to Geo altitude[4] with a period of 12 hours. Molniya type orbits, and other elliptical patterns as well, can be obtained within the Flower Constellation framework; this type of orbits is typically used for telecommunications.

[4]Often referred to as apogee dwell

The *Ellipso* constellation [9, 10] proposes the use of elliptical orbits in order to provide coverage of specific areas of interests. *Ellipso* is made of two orbits on the equatorial plane and two orbits critically inclined ($i = 63.4$ deg) to provide coverage of the northern latitudes. The use of elliptical orbits constellations, encouraged by J. Draim also in [11, 12, 13], makes the spacecrafts more expensive since they must cope with radiations in the Van Allen belt, varying range and angular size of the target area, varying in track and relative position of satellites in the same orbit [8]. The *Ellipso* constellation however has been designed by successive refinements, without the help of a comprehensive framework like the Flower Constellations; how to generate similar MEO elliptical constellations within the framework is shown in this work.

Given the many options and often competing factors involved in satellite constellation design, the use of stochastic global optimization techniques, such as Genetic Algorithms (GA), has been explored by recent research. A multi-objective optimization GA has been used in [14] to address the issue of optimal orbital altitude versus optimal number of satellites for continuous Earth coverage. The approach was, however, limited to circular orbits, i.e. Walker Delta Patterns, and issues of interfacing with existing orbital analysis software, AGI STK, limited the study to few configurations. In [15] a parallel GA is utilized for the optimization of *Ellipso* orbits; only specific cases are addressed however, whereas this dissertation proposes a more comprehensive framework for satellite constellations design. Another proposed attempt to constellation design using GAs can be found in [16].

One particularly attractive factor for Walker constellations is that they can be described by few parameters (3 integer and 2 real numbers) and, therefore, there is a finite number of them suitable for a given mission profile. Examining all configurations and finding those that provide the best mission performances is possible in the era of digital computers and parallel processing. This characteristic is shared also by Flower

Constellations and, once some assumption on the type of desired configurations are made, an exhaustive search of the optimal solution can be performed. This type of brute force search would greatly benefit by the use of parallel computing. Algorithms have been developed to greatly reduce the number of interesting configurations, for the case of symmetric Flower Constellations.

Chapter II of this dissertation briefly introduces the basic concepts of orbital mechanics required to understand the theoretical background on Flower Constellations and introduces a simple method for very fast propagation using Kepler's equation. It also explain the perturbation model used for FC design.

Chapter III shows the principles that generated the Flower Constellations idea, what is the meaning of each of the design parameters, and describes in more details which properties are shared between the spacecrafts belonging to the same constellation. A short introduction to some basic elements of geodesy and coordinate systems is also included in the same chapter.

Chapter IV introduces global optimization and the techniques utilized to solve this problem in the context of constellation design. In particular it describes the general algorithms used for this purpose, such as genetic algorithms. It also comments on how constraints between the optimization parameters can be handled with genetic algorithms.

An important part of the development of the Flower Constellations, both in time required and results obtained, is the FCVAT program [17]: this application, by allowing a quick visual interaction with the design parameters, showed the potential of the Flower Constellations and helped the theoretical developments since the early stages. The FCVAT application is complemented by the FCToolbox which is a set of programs and classes in Matlab that allow efficient analysis on Flower Constellations by interfacing with the output of the FCVAT program or providing input configurations

to it.

Chapter V describes how the problems entailed in the implementation of the proposed constellation design techniques have been faced and resolved. In particular, how to handle constraints involving several of the parameters describing a Flower Constellation in the global optimization algorithms is described. Since this dissertation is aimed at addressing the design issues of Flower Constellations, some specific examples aimed at solving specific missions such as global coverage, reconnaissance, regional coverage, and global navigation are examined in chapter VI.

Finally, in chapter VII, the results are summarized and discussed to highlight both positive aspects and limitations, thereby laying the foundations for a continuation and improvement of the present work.

CHAPTER II

ORBITAL MECHANICS BACKGROUND

This chapter introduces the background necessary to understand the developments in the rest of the dissertation. A brief introduction to orbital mechanics is deemed necessary; this subject, however, is presented here in a very concise format with the main purpose of defining terminology and clarifying notation, since there are excellent and much deeper treatments of the subject by Battin [18], Vallado [19], Schaub and Junkins [20], just to name a few, readily available in literature.

A. Orbital Mechanics Overview

The foundations of orbital mechanics are the subject of a very fascinating and long history interlocked with the development of Einstein's theory of General Relativity, going back to the classical mechanics of Isaac Newton, and even earlier to the work of Galileo Galilei or, if one wants, even further back to the great Greek philosophers, like Aristotle himself. This thesis is not the most appropriate occasion to tell this long story and the references already mentioned, especially Battin [18] and Vallado [19], do a good service in explaining the main events.

The most relevant equations for the Flower Constellation theory are mainly concerned with two body Keplerian orbital mechanics and the main perturbation term of the Earth gravitational field spherical harmonics expansion or, as is commonly referred to, the J_2 effect, which will be explained in the following sections.

Keplerian orbital mechanics is the approximation of the satellite orbits with conics, in which the central body occupies one of the foci.[1] This simpler approximation

[1]From Latin, *focus* which means *fire*, since planets orbit the Sun.

is more than accurate enough for studying general characteristics of satellite constellations, whereas a more accurate and expensive analysis is performed only on the configurations of greater interest.

Keplerian orbital mechanics is essentially based on Kepler's three laws:

1. *Planets revolve around the Sun following elliptical orbits of which the Sun occupies one of the two foci.*

2. *The radius vector from the Sun to each planet sweeps out equals areas in equal time.*

3. *The square of the period of the planet orbit is proportional to the cube of the semi major axis of the orbital ellipse.*

These laws, published by Johannes Kepler (1571-1630) in the span of several years, derived using Tycho Brahe's observations of Mars, have been later explained comprehensively by Isaac Newton in the *Philosophiae Naturalis Principia Mathematica* with the fundamental result of unifying the laws of mechanics on Earth with the laws that govern the motion of celestial bodies on a cosmic scale. Newton's law of universal gravitation in fact is remarkably simple:

$$\vec{F}_{1,2} = G\frac{m_1 m_2}{r^3}\vec{r} \tag{2.1}$$

in which $G = 6.67300 \times 10^{-11}\mathrm{m^3kg^{-1}s^{-2}}$ is the constant of universal gravitation, m_1 and m_2 are the masses of the two bodies involved and $\vec{r}$ is the vector from the center of mass of body 1 to the center of mass of body 2, and, finally, $\vec{F}$ is the gravitational force exerted between the two bodies. The resulting second order, non linear differential equation governing the motion is then:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} \tag{2.2}$$

In which $\mu = G(m_1 + m_2)$ is the planetary gravitational constant, and usually, the satellite mass m_2 being much smaller than the planet mass m_1, it is approximated with $\mu = Gm_1$; thus the planetary gravitational constant is independent from the satellite mass; for Earth, $\mu = 398600.441\text{km}^3/\text{s}^2$.

The study of analytical methods to solve this seemingly innocent second order differential equation have occupied many papers and the minds of some of the most brilliant mathematicians for several centuries, receiving contributions from Euler, Gauss, and Lagrange, among many others. Solutions in power series expansion, the F and G functions solution, universal functions, etc. have been devised , but these will not be described here, refer to [18] for a comprehensive treatment. The classical elliptical geometric solution instead is chosen since it is the more natural starting point to develop the Flower Constellation theory. The following developments briefly summarize the presentation of the same material that can be found in [21].

It is important to emphasize the assumptions and conditions under which the following development remains valid: the two body problem is concerned with studying the motion of one body, i.e. a satellite, orbiting a much larger mass, i.e. a planet, without the interference of other bodies or other disturbance forces (like atmospheric drag or solar pressure, for instance). In the vicinity of Earth, or of any other major planet, such approximation remain valid and remarkably accurate within the so called *Sphere of Influence* (SOI) of the planet; the SOI is rather more a useful abstraction than a physical reality and for Earth its radius is roughly $R_{SOI} = 1.5 \times 10^6 \quad \text{km} = 0.01\text{AU}$, (1AU or astronomical unit is the mean distance between Earth and Sun.

Conservation of Angular Momentum

Second Kepler's law can be easily proven by showing that eq. (2.2) implies conservation of the angular momentum. By taking the cross product of eq. (2.2) with the

position vector $\vec{r}$ we get:

$$\vec{r} \times \ddot{\vec{r}} = \frac{d}{dt}(\vec{r} \times \vec{v}) = 0 \tag{2.3}$$

Which proves that the quantity:

$$\vec{h} = \vec{r} \times \vec{v} \tag{2.4}$$

defined as the angular momentum per unit mass, is a constant. Hence the motion happens on a plane, and by using the polar coordinates expression

$$h = r^2 \dot{\varphi}\, \hat{i}_z \tag{2.5}$$

where $\hat{i}_z$ is a vector perpendicular to the $\vec{r}, \vec{v}$ plane. Since the rate at which the radius vector sweeps out area is $1/2\, r^2 \dot{\varphi}$ this also proves Kepler's second law.

Energy Integral

From the *eccentricity vector*

$$\mu \vec{e} = \vec{v} \times \vec{h} - \frac{\mu}{r} \vec{r} \tag{2.6}$$

which is another constant of the motion[2], one can derive the expression of the constant orbital energy density. The squared magnitude of eq. (2.6) is:

$$\vec{e} \cdot \vec{e} = 1 - e^2 = \frac{h^2}{\mu} \left(\frac{2}{r} - \frac{v^2}{\mu} \right) \tag{2.7}$$

By defining $p = h^2/\mu$ as the *parameter*[3] and $a = (2/r - v^2/\mu)^{-1}$ we get also $p = a(1 - e^2)$ as the sum of kinetic and potential energy per unit mass:

$$E = \frac{v^2}{2} - \frac{\mu}{r} = -\frac{\mu}{2a} \tag{2.8}$$

---

[2] Always directed at periapsis and whose magnitude is the orbit eccentricity; sometimes it is known as Laplace vector

[3] More often called the *semilatus rectum*

which is a constant.

Polar Equation of Orbit

By taking the dot product of eq. (2.6) with the position, $\vec{r}$ we get:

$$\vec{r} \cdot \vec{e} = \frac{1}{\mu} \vec{r} \cdot \vec{v} \times \vec{h} - \frac{\vec{r} \cdot \vec{r}}{r} \tag{2.9}$$

$$\vec{r} \cdot \vec{e} = \frac{h^2}{\mu} - r \tag{2.10}$$

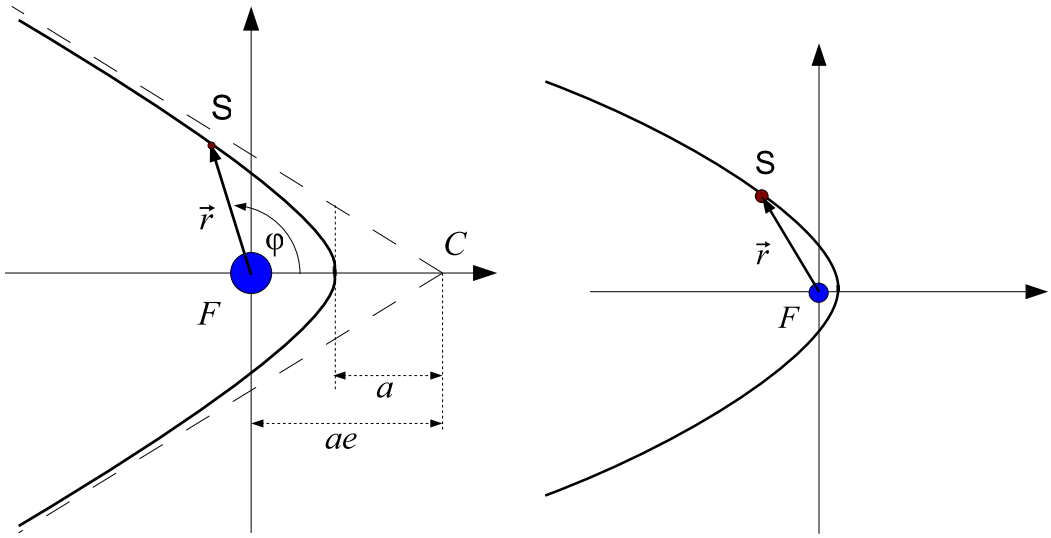
$$r(1 + e\, cos\varphi) = p \tag{2.11}$$

$$r = \frac{p}{1 + e\, cos\varphi} \tag{2.12}$$

In (2.12) the *true anomaly*, φ, is the angle between the eccentricity vector $\vec{e}$ and the radius vector $\vec{r}$. The orbit is symmetrical with respect to the axis defined by the vector $\vec{e}$ and it is bounded if $e < 1$, unbounded if $e > 1$.

The resulting conic may be an *ellipse*, a *parabola* or a *hyperbola*, as can be visualized by thinking to the intersection of a cone and a plane at different angles; the type of the conic is determined by the constant a, called semi-major axis: the orbit is an ellipse if $a > 0$, an hyperbola for $a < 0$ and a parabola if $a \to \infty$; therefore if the associated energy (per unit mass), $E = -\mu/2a$, is negative then the orbit is an ellipse, if the energy is positive than it is a hyperbola, if it is 0, then the orbit is parabolic. Fig. 1(a) and 1(b) show the geometry of hyperbolic and parabolic orbits respectively. A geometrical definition of some of the quantities introduced in this section is given in Fig. 2, for elliptical orbits.

Although the laws of Newtonian mechanics have been incorporated within the larger framework of Einstein's *General Relativity* they still prove accurate and valuable to within the precision usually required for celestial navigation.



(a) Hyperbolic orbit (b) Parabolic orbit

Fig. 1. Hyperbolic and parabolic orbit geometry.

Kepler's Equation

As it has been briefly shown, the satellite position $\vec{r}(t)$ depends on time through the *True Anomaly* angle φ. Finding φ from time is a non-trivial problem and it motivated the development of a vast literature of numerical and analytical methods. Elliptical integral of the first kind have been introduced trying to solve analytically eq. (2.13), which can be easily derived from Eq. (2.5) and eq. (2.12).

$$\sqrt{\frac{\mu}{p^3}}dt = \frac{d\varphi}{1 + e\,\cos\varphi} \tag{2.13}$$

A much easier alternative is that of solving Kepler's equation (KE), (2.14), which relates time to *Eccentric Anomaly*, E.

$$\sqrt{\frac{\mu}{a^3}}(t - t_p) = E - e\sin E \tag{2.14}$$

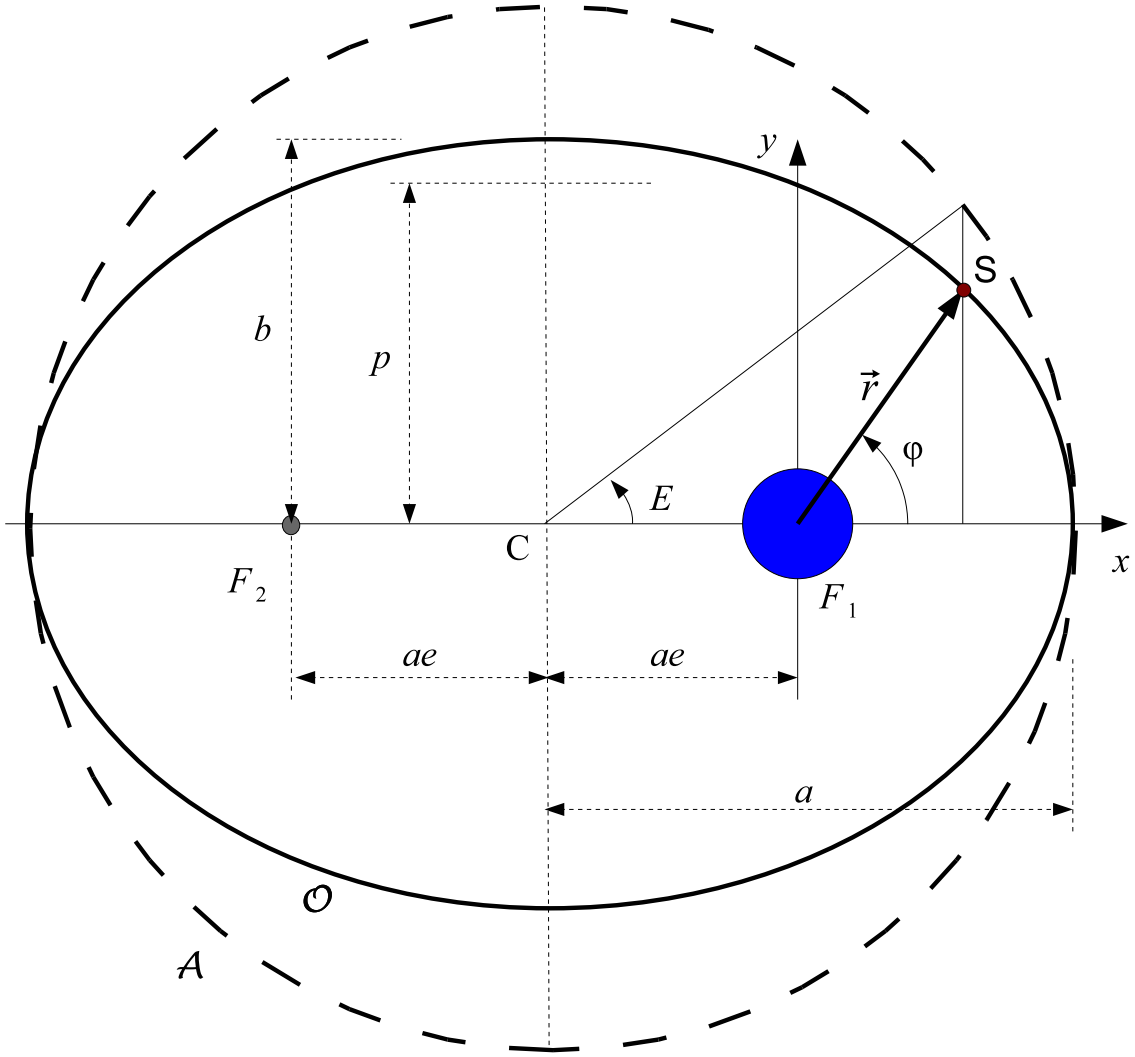


Fig. 2. Elliptical orbit geometry. The satellite S is orbiting a planet on an elliptical orbit $\mathcal{O}$ of semi major axis a and eccentricity e. The planet is occupying one of the foci, F_1 and the auxiliary circle $\mathcal{A}$ of radius a is used to geometrically define the *Eccentric Anomaly*, E. The position of the satellite at any instant is defined by the radius vector $\vec{r}$ which is a function of the *True Anomaly*, φ.

In which t_p is the time of passage at pericenter. This relation can be easily derived by using eq. (2.5) and the fact that $r\,d\varphi = b\,dE$. The left hand side of eq. (2.14) is called *Mean Anomaly*, M, and it is an angle variable proportional to time t, with constant angular velocity $n = \mu/a^3$, a quantity called *Mean Motion*. Thus the most usual form of Kepler's equation is that of eq. (2.15).

$$M = E - e\sin E \tag{2.15}$$

Eq. (2.15) is a transcendental equation that can be solved numerically for E. The value of the *True Anomaly*, φ, can then be recovered using eq. (2.16).

$$\tan\frac{\varphi}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \tag{2.16}$$

Orbital Elements

A dynamical system described by eq. (2.2) is completely determined if at any instant position and velocity (i.e. the state) of the satellite are given. Since position and velocity are three dimensional vectors this means that 6 parameters are needed to completely identify the system state at any given time. The choice of the parameters is however arbitrary and other integrals of the motion can be used as well. One of the most well known representation is the *Classical Orbital Elements*, which is a set of six parameters defining the shape of the orbit through: 1) the semi major axis a and 2) eccentricity e; the orientation of the orbital plane through the angles: 3) i for *inclination*, 4) Ω for the *Right Ascension of Ascending Node* (RAAN), and 5) ω for the *Argument of Perigee*[4]. Finally the 6^{th} parameter, position of the satellite on the orbit, is given by the $MeanAnomaly$ angle described in eq. (2.15).

[4]Argument of periapsis, in general

B. Avoid Solving Kepler's Equation

Computing the spacecraft trajectory for a desired period of time is often necessary during mission analysis or, more generally, spacecraft orbit simulation. The classical analytical approach for addressing this problem requires the solution of KE, by using iterative numerical methods.

The concept presented here shifts the focus from the issue of an efficient and accurate solution to KE towards a completely different approach in which such solution is not required at all, at least for some applications. The proposed method is particularly beneficial when the evaluation of the position of a spacecraft at precise instants of time is not required.

An approximation of the spacecraft trajectory is often appropriate for the task at hand. Such applications may include graphic visualization of the trajectory, coarse estimation of spacecraft rise and set time, and coverage analysis for communication or navigation. In these applications it is usually inconsequential whether the sampling of the trajectory is done at some very precise, uniform intervals of time Δt or at a somewhat variable rate, as long as a given level of accuracy in the analysis is maintained.

Solving the rise and set problem has been indeed the need that generated interest in the subject of this section: how one can compute efficiently an approximated trajectory for the satellites so that hundreds of candidate constellation configurations may be examined by a Genetic Algorithm quickly?

By profiling the code used to solve the raise and set problem it was found that more than 90% of the computation time was being spent in solving Kepler's Equation, therefore alternatives have been sought.

Analytical orbit propagation (e.g., two-body or linear J_2) requires solving KE

at any time instant in which the orbital position knowledge is needed. Analytical orbit propagation is extremely useful to speed up analysis and to avoid numerical integration of the equations of motion. For this reason KE has been deeply studied since it appeared about 350 years ago. Since then, many methods to solve KE have been proposed.

The classical methods are based on iterative procedures, on truncated series expansions, or on both. References [18, 22, 19] contain detailed reviews of these historical methods, while some more efficient methods have been proposed by Nijenhuis [23] and Markley [24]. More recently Fukushima [25] and then Feinstein and McLaughlin [26], developed the presently fastest algorithms using discretization techniques that require large tables of pre-computed data. A method to solve KE which take advantage from the great flexibility of Bézier curves, has been recently proposed in [27]. All this excellent work however focuses on high accuracy and code efficiency. If hundreds of configurations of constellations must be examined by a Genetic Algorithm or by an exhaustive search, then one would like to have something even simpler than the celebrated Newton-Raphson method:

$$M_k = E_k - e\ \sin E_k \tag{2.17}$$

$$E_{k+1} = E_k + \frac{M - M_k}{1 - e\ \cos E_k} \tag{2.18}$$

where M is the value of the true anomaly in input, the starting point $E_0 = M$ is used as starting guess. Iteration stops when the desired accuracy has been reached, i.e. $|E_{k+1} - E_k| < \epsilon$.

The transformation from true anomaly, φ, to ECI position is:

$$\vec{r} = s\frac{p}{1 + e\,\cos\varphi}\,R_{IO}\begin{bmatrix}\cos\varphi\\ \sin\varphi\\ 0\end{bmatrix} \quad \text{and} \quad \vec{v} = \sqrt{\frac{\mu}{p}}\,R_{IO}\begin{bmatrix}-\sin\varphi\\ e + \cos\varphi\\ 0\end{bmatrix} \tag{2.19}$$

and the transformation from time to mean anomaly is simply:

$$M = n\,(t - t_p) \tag{2.20}$$

In short, the orbit propagation problem can be seen as transformations between angles:

$$t \overset{(2.20)}{\longrightarrow} M \overset{(2.15)}{\rightsquigarrow} E \overset{(2.16)}{\longrightarrow} \varphi \overset{(2.19)}{\longrightarrow} \vec{r} \tag{2.21}$$

where the equations used in each transformation are indicated. The slowest part of the procedure is represented by the step $M \overset{(2.15)}{\rightsquigarrow} E$, requiring the solution of Kepler's Equation.

Typical Propagation Algorithm

The usual approach to calculating a spacecraft orbit involves the computation of the spacecraft position vector $\vec{r}(k\Delta t)$ for $k = 0 \ldots N$, and $\Delta t = T/N$, where $N > 0$ is the number of discrete time intervals to be used in the trajectory prediction. By assuming elliptical motion of the spacecraft, algorithm 1 describe a simple analytical approach to computing the spacecraft position.

Algorithm 1 is used to compute the satellite position in the orbital reference frame, then the coordinate transformation from orbital to inertial frame, R_{IO}, is computed and applied to obtain the spacecraft position, $\vec{r}$, in the Earth Centered Inertial (ECI) reference frame. The most time consuming step is step 2: solving KE.

The spacecraft position at a specific time can be used, for instance, to compute

Algorithm 1 PROPAGATION(t, t_p, ORBEL)

Require: A time t, time of perigee passage t_p, ORBEL $= (a, e, i, \Omega, \omega, M_0)$ Orbital elements at time t_0

Ensure: $\vec{r}(t)$ spacecraft position at time t

1: $M \leftarrow n(t - t_p)$ {compute mean anomaly}

2: $E \leftarrow$ KEPLER(M, e) {compute eccentric anomaly}

3: Compute true anomaly, φ using Eq. (2.16)

4: Compute position $\vec{r}(t)$ using Eq. (2.19)

5: **return** $\vec{r}(t)$

the visibility of the satellite from a ground station (or viceversa). If this is the case algorithm 1 is executed $N + 1$ times, for $t = t_0 \dots t_N$.

Without Kepler's Equation

The need for solving KE can be removed by discretizing the eccentric anomaly rather than the mean anomaly (i.e. time).

The transformation sequence given in Eq. (2.21) changes into the following:

$$t \overset{(2.20)}{\longleftarrow} M \overset{(2.15)}{\longleftarrow} E \overset{(2.16)}{\longrightarrow} \varphi \overset{(2.19)}{\longrightarrow} \vec{r} \tag{2.22}$$

or to the sequence:

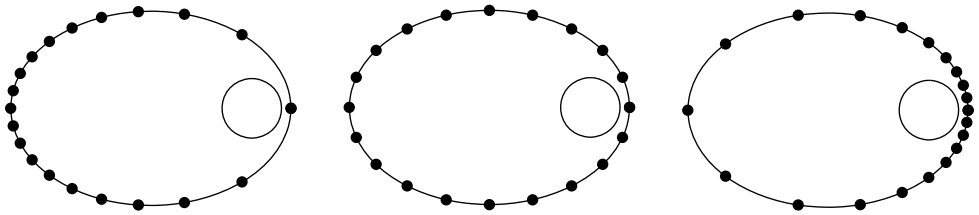
$$t \overset{(2.20)}{\longleftarrow} M \overset{(2.15)}{\longleftarrow} E \overset{(2.16)}{\longleftarrow} \varphi \overset{(2.19)}{\longrightarrow} \vec{r} \tag{2.23}$$

depending on whether we want to discretize E or φ, respectively.

It is clear that, by choosing to discretize E or φ, we do not obtain a constant time step. However, most of the times, the non uniformity of the time step, which becomes worse as the orbit eccentricity increases, is not a drawback: in fact it may

have positive aspects; using constant ΔE (or $\Delta\varphi$) increments, results in having more samples where the satellite velocity is higher (i.e. at perigee passages). Therefore, in such applications where experiments and observations are taken at perigee, a constant ΔE (or $\Delta\varphi$) step approach presents advantages. This insight apply to elliptical orbits only, since the three angles are the same for circular orbits.

In some cases, we can get away without bothering about time at all: one could discretize the angles (i.e. *anomalies*) instead, and obtain other benefits. Figures



(a) Mean anomaly step (b) Eccentric anomaly step (c) True anomaly step

Fig. 3. 20-point orbit propagation using different angle steps.

3(a), 3(b), and 3(c), show the orbital locations by propagating with constant mean, eccentric, and true anomaly, respectively. These figures highlight, even better than equations, the following facts:

1. If one is interested in the perigee passages, then use constant true anomaly step; there is no need to solve KE.

2. If one is not interested in a specific orbit section, then propagation with constant eccentric anomaly step can be used; there is no need to solve KE.

3. If one is interested in the apogee or in specific time instants, then propagate using constant mean anomaly (time) step, solving KE. If high accuracy is not needed, then use the simplified algorithms proposed in the next sections.

Linear and Quadratic Propagations

When a constant time step is desired and very high accuracy is not needed, it is possible to use a numerical artifice that provides a quasi-constant time-step sequence. The outline of the method is provided in the following.

Let us consider the eccentric anomaly discretization: what we want to do is to choose a sequence of E_k such that the associated sequence of mean anomaly M_k (i.e. times t_k) is such that $(M_k - M_{k-1})$ is equal to an assigned constant difference ΔM, independent from the index k. In order to obtain an approximated sequence we differentiate eq. (2.15) we obtain:

$$dE = \frac{dM}{1 - e\,\cos E} \tag{2.24}$$

If one rewrites eq. (2.24) for finite but small differences:

$$\Delta E_k = E_{k+1} - E_k = \frac{\Delta M}{1 - e\,\cos E_k} \tag{2.25}$$

that is

$$E_{k+1} = E_k + \frac{\Delta M}{1 - e\,\cos E_k} \tag{2.26}$$

where the ΔM value depends on the desired number of points per orbit N:

$$\Delta M = \frac{2\,\pi}{N} \tag{2.27}$$

A second order Taylor expansion yields an alternative method that provides a better accuracy at the cost of a small increase in complexity:

$$M_{k+1} = E_k - e\,\sin E_k + (1 - e\,\cos E_k)\,\Delta E_k + \frac{1}{2}\,e\,\sin E_k\,(\Delta E_k)^2 \tag{2.28}$$

therefore

$$\Delta M = (1 - e\,\cos E_k)\,\Delta E_k + \frac{1}{2}\,e\,\sin E_k\,(\Delta E_k)^2 \tag{2.29}$$

Solving for ΔE_k we obtain the second-order solution for quasi constant-time propagation

$$E_{k+1} = E_k + \frac{1}{2}\sqrt{1 + \frac{2\,\Delta M\, e\, \sin E_k}{(1 - e\, \cos E_k)^2}} - \frac{1}{2} \tag{2.30}$$

Numerical Tests

How well this simple method behave? Can we compare it with the classical Newton-Raphson approach? Newton-Raphson is by far the most widely used Kepler's equation solver[5]. The Newton-Raphson method is used everywhere because it is simple to code and it is efficient enough for most practical purposes. The algorithm proposed above is even simpler, and therefore there is hope for expecting that it might be utilized too.

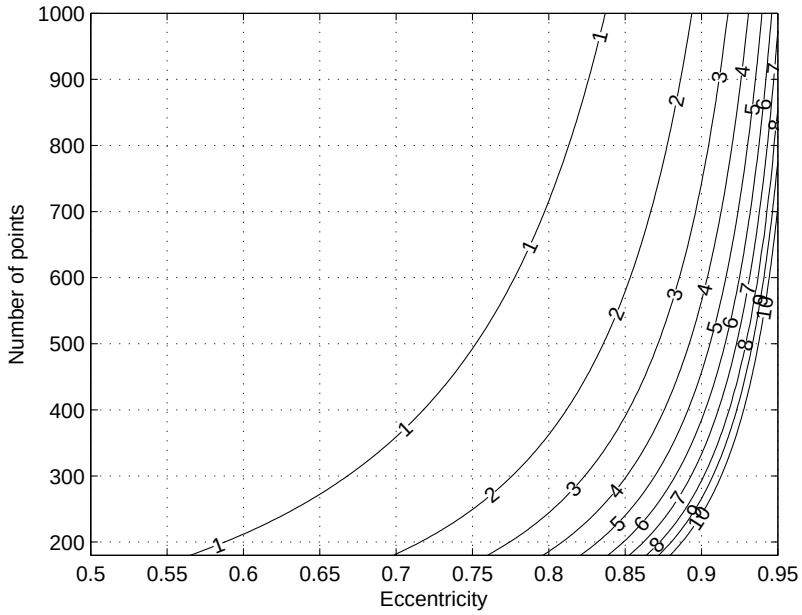
A straightforward accuracy comparison of the two methods is not appropriate since they are logically very different in purpose: the idea presented here in fact tries not to solve KE at all, whereas the Newton-Raphson algorithm is a Kepler equation solver. For this reason, two measures have been adopted to characterize the accuracy of the proposed algorithms, assuming a propagation for half orbit as a function of eccentricity and number of sample points, N:

1. How close is the last item in the sequence of computed eccentric anomaly to the expected value of 180° ?

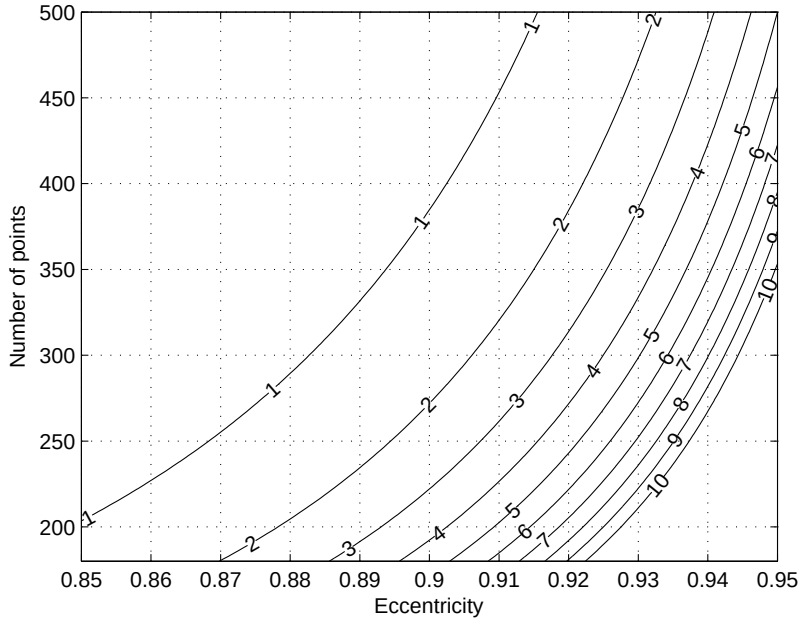
2. How constant is the actual ΔM obtained from the E sequence output by the two algorithms?

Numerical tests have been performed using MATLAB and the results are summarized in the Figures 4, 5, 6. The tests show the behavior of the two methods with

[5]A survey of the code available on the internet can easily convince any skeptic about the validity of this statement.

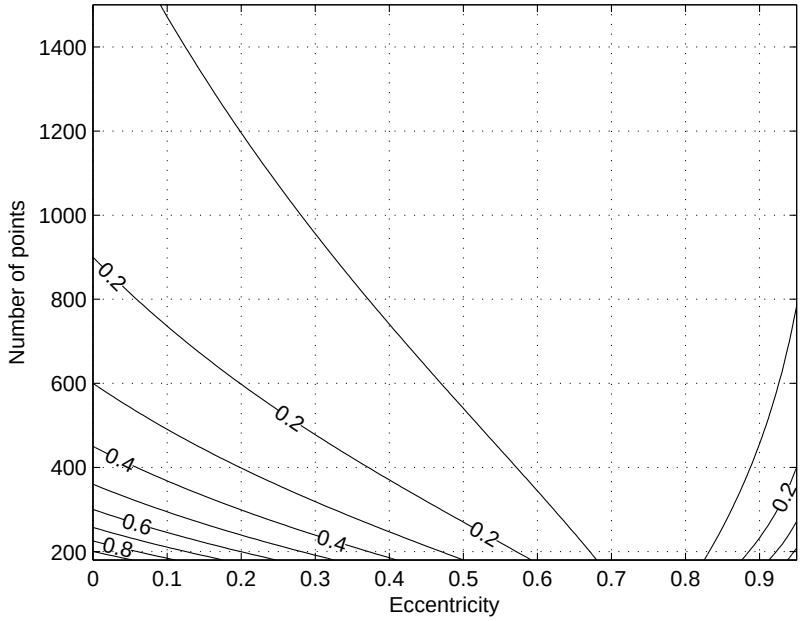


(a) First order method.

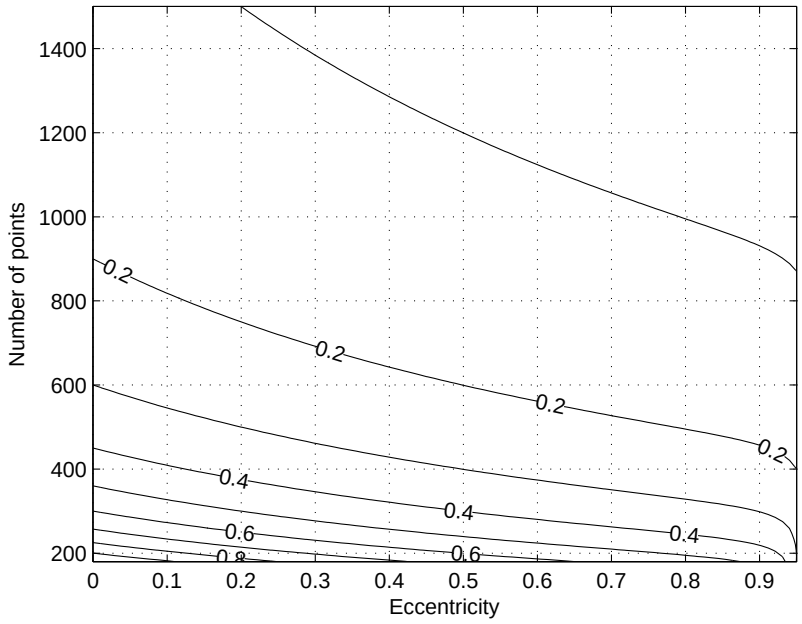


(b) Second order method.

Fig. 4. Relative percentage variation of ΔM between two time steps.

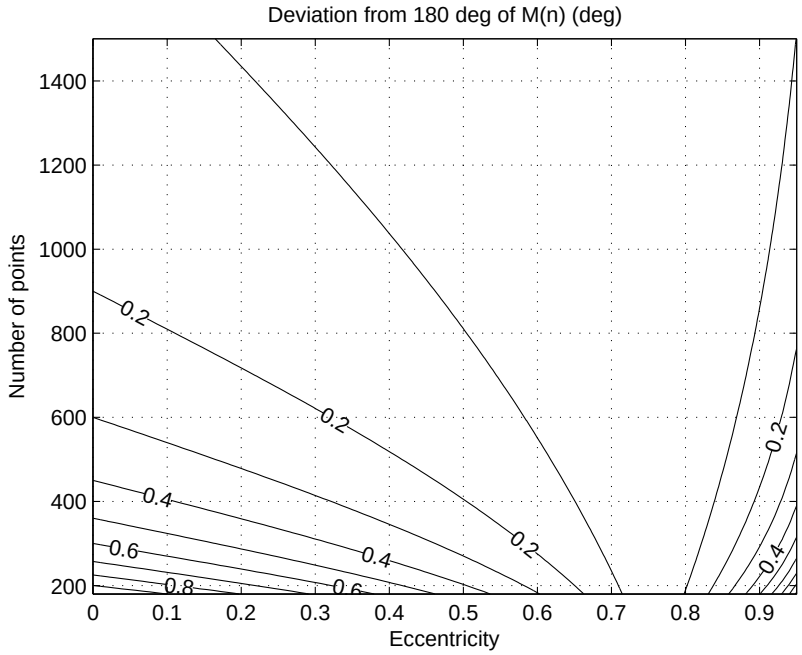


(a) First order method.

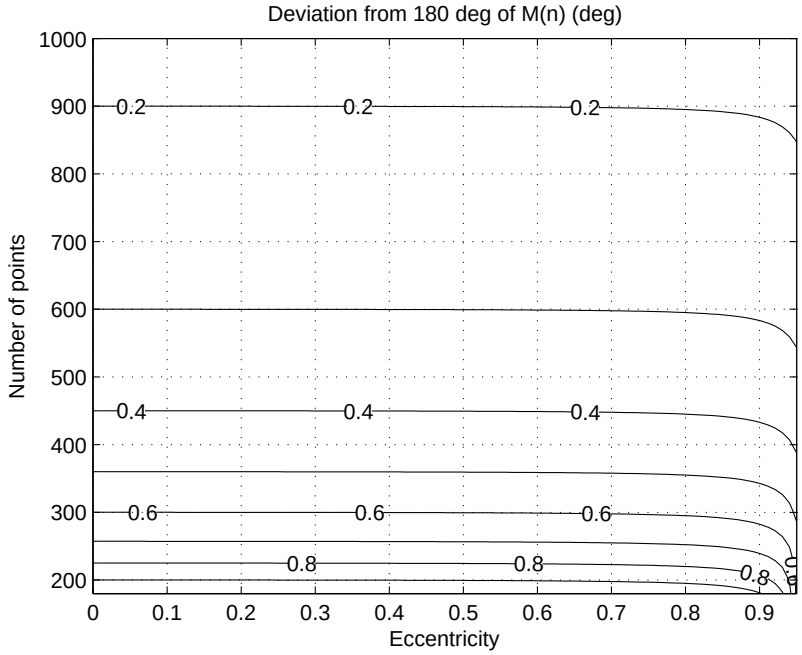


(b) Second order method.

Fig. 5. Error in eccentric anomaly, E at last step, as function of e and N.



(a) First order method.



(b) Second order method.

Fig. 6. Error in mean anomaly, M at last step, as function of e and N.

respect to the number of points, N, and the increasing value of eccentricity. We emphasize that these methods are intended for those applications in which "What time is it?" can be answered approximately, as long as the orbit is computed accurately as a function of E or φ.

Summarizing, the first order method already proves to be accurate enough for orbits of moderate eccentricity (the vast majority of the practical cases), whereas the second order method is much more accurate, but the increased number of operations makes it less appealing.

C. Orbital Perturbations

Eq. (2.2) is valid with the simplificative assumption that there are no other disturbance forces acting on the two bodies and that the point mass approximation holds. In reality both these assumptions are not satisfied, therefore one way to correct eq. (2.2) is that of introducing a disturbance acceleration $\vec{a}_d$:

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} + \vec{a}_d \tag{2.31}$$

with this model the main dynamic is still captured by an equation of the same form as Eq. (2.2) and only a disturbance term $\vec{a}_d$ is added to take into account all the sources of disturbance.

The main sources of orbital perturbations are:

- Atmospheric drag below 600 Km of altitude, i.e. Low Earth Orbit, or LEO for short.

- Equatorial bulge effect[6]: Earth is not a sphere but an ellipsoid, thus by expand-

[6]Also frequently called J_2 effect

ing the gravitational potential in spherical harmonic series[7], it can be shown that the higher order terms cause disturbance forces [20].

- Gravitational effects from other, more distant, bodies.
- Solar pressure, i.e. the effect of radiation and particles from the solar wind hitting the spacecraft.

If atmospheric drag is excluded, by selecting an appropriate altitude, and the solar pressure effects are neglected, since they are very small in general, then the other components can be expressed as a disturbing potential $R(\vec{r})$:

$$\vec{a}_d = \frac{\partial R}{\partial \vec{r}} \tag{2.32}$$

The effect of the perturbations on the orbital elements has been studied extensively, producing analytical, semi-analytical and numerical solutions to Eq. (2.31).

The most widely known analytical expressions in terms of the orbital elements have been derived by Lagrange and Gauss, using the technique of *variation of parameters* [28]. Lagrange's form is reported in Eq. (2.33).

$$\dot{\Omega} = \frac{1}{nab\sin i}\frac{\partial R}{\partial i} \tag{2.33a}$$

$$\dot{i} = -\frac{1}{nab\sin i}\frac{\partial R}{\partial \Omega} + \frac{\cos i}{nab\sin i}\frac{\partial R}{\partial \omega} \tag{2.33b}$$

$$\dot{\omega} = -\frac{\cos i}{nab\sin i}\frac{\partial R}{\partial i} + \frac{b}{na^3 e}\frac{\partial R}{\partial e} \tag{2.33c}$$

$$\dot{a} = \frac{2}{na}\frac{\partial R}{\partial \lambda} \tag{2.33d}$$

$$\dot{e} = -\frac{b}{na^3 e}\frac{\partial R}{\partial \omega} + \frac{b^2}{na^4 e}\frac{\partial R}{\partial \lambda} \tag{2.33e}$$

$$\dot{\lambda} = -\frac{2}{na}\frac{\partial R}{\partial a} - \frac{b^2}{na^4 e}\frac{\partial R}{\partial e} \tag{2.33f}$$

[7]The most significative coefficient in the expansion is called $J_2 \cong 0.0010826269$, hence the name.

where $\lambda = -M_0 = -n\,t_p$, and t_p is the time of passage at pericenter.

If the disturbing potential R is assumed to be caused only by the J_2 term in the Earth's gravitational potential field, R assumes the form of eq. (2.34).

$$R = -\frac{Gm}{r} J_2 \left(\frac{R_e}{r}\right) P_2(\cos\phi) \tag{2.34}$$

in which $R_e = 6378.137\ Km$ is the Earth equatorial radius, according to the WGS84 reference ellipsoid, and $P_2(\cos\phi)$ is the expression in (2.35):

$$P_2(\cos\phi) = \frac{1}{2}[3\,sin^2(\omega+\varphi)\,\sin^2 i - 1] \tag{2.35}$$

By taking the average over one orbit of the disturbing potential, eq. 2.33 can be greatly simplified at the price of disregarding smaller periodic disturbances and only taking into account the secular terms, i.e. the linear terms. Eq. (2.36) gives the expression of the averaged potential, $\overline{R}$.

$$\overline{R} = \frac{1}{2\pi}\int_0^{2\pi} R\,dM = \frac{n^2 J_2 R_e^2}{4(1-e^2)^{3/2}}(2 - 3\sin^2 i) \tag{2.36}$$

By substituting eq. (2.36) into (2.33) and simplifying one gets equations (2.37).

$$\overline{\dot{a}} = 0 \tag{2.37a}$$

$$\overline{\dot{i}} = 0 \tag{2.37b}$$

$$\overline{\dot{e}} = 0 \tag{2.37c}$$

$$\overline{\dot{\Omega}} = -\frac{3}{2} J_2 \left(\frac{R_e}{p}\right)^2 \overline{n}\cos i \tag{2.37d}$$

$$\overline{\dot{\omega}} = -\frac{3}{4} J_2 \left(\frac{R_e}{p}\right)^2 \overline{n}(5\cos^2 i - 1) \tag{2.37e}$$

$$\overline{\dot{M}} = \overline{n} = n\left[1 + \frac{3}{4} J_2 \left(\frac{R_e}{p}\right)^2 (2 - 3\sin^2 i)\sqrt{(1-e^2)}\right] \tag{2.37f}$$

From equations (2.37) it becomes apparent that the secular effects of the J_2

perturbation cause: a) precession of the line of nodes Ω, b) precession of the line of apsides ω, and c) change in the orbital period since $T = 2\,\pi/\overline{n}$. There exists a critical inclination for which the line of apsides perturbation (2.37e) vanishes, this inclination for Earth is $i = 63.4^\circ$ or $i = 116.6^\circ$.

In this dissertation only the J_2 perturbation effect is considered because one can always choose orbits that avoid the atmospheric drag problem, whereas the effect of other bodies is negligible since it is assumed that the constellations are placed well within Earth SOI. Another reason for not considering higher order perturbations is that existing numerical and semi-analytical propagators, although very accurate, need more computational resources than a simple J_2 approach and the degree of precision obtainable with this scheme is accurate enough to account for the most important perturbation effect. Of course, all modeling errors must, ultimately, be accommodated through orbit station keeping. This issue is considered, but to a secondary degree, in this dissertation.

CHAPTER III

FLOWER CONSTELLATION BACKGROUND

This chapter presents background material regarding constellation design in general and Flower Constellations in particular, summarizing the relevant literature presented by Mortari, Wilkins and Bruccoleri in [29], with the most recent additions by Mortari and Avendano [30].

A. Introduction to Constellation Design

Constellations of satellites have been proposed and realized to perform a number of missions that fall into three broad classes: Earth Observation, Navigation, and Telecommunications. The success or failure of a satellite constellation project is not only measured by technical achievements but also by its economic benefits, since the costs of building and operating satellite constellations are enormous. Fig. 7 shows the main cost-driving factors for a generic constellation design. This dissertation is concerned mainly with the top level design issues and with system performance evaluation. The issues regarding the launch vehicle, how the spacecraft assembly and production line are organized, building and integration of sensors, the ground segment, etc. will not be addressed here. It is only briefly mentioned, however, that the production of spacecrafts for a constellation allows for economy of scale since a number of identical satellites must be built and therefore the time required for the production of one unit can be greatly reduced thanks to the possibility of establishing an assembly chain.

J.G. Walker worked for the Royal Aircraft Establishment (UK) and published a series of papers [6, 7] from 1971 to 1984, regarding the design of satellite constellations. One of the most successful is the simple *Delta Pattern* which is completely

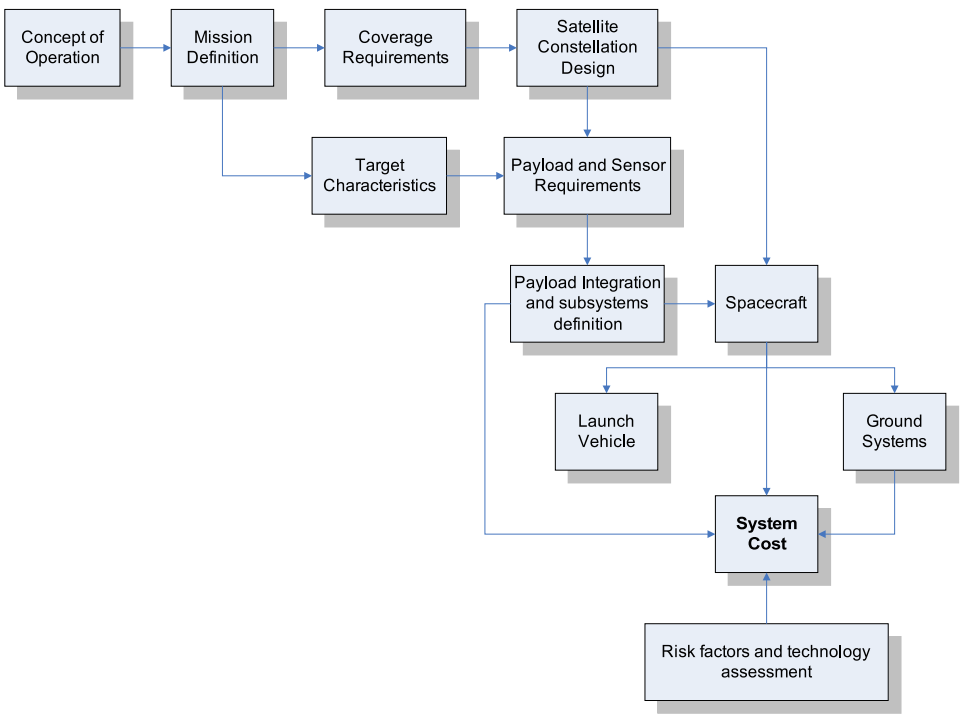


Fig. 7. Major cost drivers in mission design.

characterized by only 5 parameters. The first three are positive integers: t, the total number of satellites p, the number of orbital planes, and f an inter-plane phasing parameter. Walker delta patterns emphasize symmetry thus there are $q = t/p$ uniformly distributed satellites per plane, from which it is clear that t must be a multiple of p for some integer q. The ascending nodes of the orbital planes are uniformly distributed according to $\Omega_k = k\,2\,\pi/p$. In order to achieve symmetry corresponding satellites in different planes must have a constant phase difference: therefore one may define a pattern angle: $\chi = 2\pi\,/\,t$ and let the phase difference between satellites on different orbital planes be $f\,\chi$, with $f = 0..(p-1)$. The other two parameters are the altitude, h, and the inclination of the planes, i (same for all planes) with respect to the equatorial plane. Since Walker constellations are only defined for circular orbits ($e = 0$), the tuple $< t, p, f, i, h >$ uniquely identify the set of orbits and satellites

orbital elements for a Walker Constellation. Algorithm 2 can be used to determine the orbital elements of the satellites in a Walker Delta pattern. It has been implemented in Matlab and utilized for comparison of Flower Constellations and other known constellations. Having explained Walker constellations is also propaedeutical to understanding Flower Constellation phasing algorithms.

Algorithm 2 WALKERDELTA$(t, p, f, h, incl)$

Require: $t \in \mathbb{N}$, number of satellites, $p \in \mathbb{N}$, number of planes, $f \in \mathbb{N}$, phase argument, $h \in \mathbb{R}$, altitude, and $incl \in \mathbb{R}$, inclination.

Ensure: $\{\mathrm{OE}_i : i = 1 \dots t\}$ Orbital Elements of all satellites in the constellation

$q \leftarrow t/p, \quad i \leftarrow 0$

$a \leftarrow h + R_\oplus, \quad e \leftarrow 0, \quad \omega \leftarrow 0$

for $k \leftarrow 1$ to p **do**

  for $j \leftarrow 1$ to q **do**

    $i \leftarrow i + 1$

    $\Omega_i \leftarrow \left(k \, \frac{2\,\pi}{p} \right) \mod 2\,\pi$

    $M_i \leftarrow \left(j \, \frac{2\,\pi}{q} + f \, \frac{2\,\pi}{t} \, k \right) \mod 2\,\pi$

    $\mathrm{OE}_i =< a, e, incl, \omega, \Omega_i, M_i >$

  end for

end for

Another well known approach to constellation design is the *Street of Coverage* (SoC) method, described in [31]. In this approach the surface of the Earth is divided in strips along the meridians and each polar orbit ($i = 90°$) is placed at the center of the strip. The orbits are designed so that the coverage of each orbit is partially overlapped with that of its neighbors and satellites are placed along each orbit to ensure continuous coverage of each street. IRIDIUM [32] is an example of a constel-

lation designed with this method, resulting in a 66 spacecraft constellation for global personal telecommunication services.

The problems with this methodology are a) the fact that there is a greater coverage of the polar regions rather than of the more inhabited equatorial region; and b) that there is an asymmetry in coverage in the strips where ascending and descending satellites are adjacent. Since coverage varies with altitude, this method of constellation design also requires circular orbits. Another serious issue related to the availability of service in SoC constellations is the fact that at any time only one satellite is potentially visible by a receiver on Earth. If for any reason, i.e. typically masking by some tall building or mountain, the satellite is not visible then the service is completely unavailable.

A direct competitor of IRIDIUM is GLOBALSTAR [33], which provides similar services and has been designed as a Walker 48/8/1 constellation, at an altitude of 1400 Km, which is high enough to avoid the effects of Earth atmosphere and the higher density of space debris, and it is below Van Allen's radiation belt [34]. IRIDIUM provides true global coverage and communications even in the open oceans or poles thanks to on-board processing and inter-satellite links, whereas GLOBALSTAR has a more limited coverage of the equatorial areas (Europe, USA, Asia, North Africa, Australia) and can not work without assistance from ground stations. Both IRIDIUM and GLOBALSTAR companies faced serious financial problems but are, at the time of this writing, still in business.

A fundamental issue for any kind of space asset, and even more so for commercial satellites, is that of the expected lifetime of spacecrafts. If for government or military missions it is conceivable to consider a spacecraft expendable, when compared with the importance of intelligence missions on which policy or military actions may depend, for commercial entities the return on the investment depends often on the number

of users and the amount of time for which the service provided by the spacecrafts is available.

Deployment and operation of satellite constellations is always risky given the enormous complexity of the ground and space segments, but what makes risk management even more critical in this case, as compared to land based aircrafts for instance, is the impossibility of correcting any mistake or component failure once the spacecrafts are in orbit (for most cases). Ways to address the inherent fragility of spacecrafts caused by lack of maintenance are:

- Redundancy of critical components;
- Minimization of moving parts;
- Extensive rigorous testing of structure, hardware, and software before launch.

However even with all the possible care failures still happen, thus the lifespan of a spacecraft is determined by the following factors:

- Mean Time Before Failure (MTBF) of critical components;
- Level of redundancy;
- Resilience to radiation (both protons and electrons);
- Fuel required for station keeping and attitude maneuvers.

The first two items are obviously related: the level of redundancy of key components is determined by the expected life time and the MTBF of critical systems (e.g. on board computers, power system, etc.). Resilience to radiation is a design requirement of the on-board electronics, but the actual amount of radiation to which the spacecraft is exposed depends on the deployment orbit. An introduction to radiation in the space environment is given in chapter VI.

B. Reference Frames

In the remainder of this dissertation some reference frames will be repeatedly used, thus it is useful to define them unambiguously in this introductory section. The only existing satellite constellation are orbiting around the Earth, therefore in this work we are concerned with reference frames that have the Earth center as origin. Concepts of interplanetary constellations have been proposed, but the changes needed to support such constellations are minimal and therefore the assumption is that a constellation is orbiting the Earth, if not otherwise stated.

The most commonly used reference frame is the Earth Centered Inertial (ECI), in which the z-axis is defined along the Earth North pole, the x-axis is defined along the *Vernal Equinox* Υ, i.e. at the intersection of the Earth equatorial plane and *Ecliptic plane*[1] during the spring equinox[2] and towards the Sun; the y-axes is chosen so that a right-handed reference system results. Although this reference system is called inertial, strictly speaking it is not truly inertial since perturbations change the direction of the axes over long periods of time; for this reason an epoch (i.e. a date) is used to define a quasi-inertial reference frame.

Since the Earth spins about its own axis with a period of 23 hours and 56 minutes, one *sidereal day*, another reference frame should be defined in which the geographical coordinates of points on the Earth do not change: such frame is the Earth Centered, Earth Fixed (ECEF) reference frame. The spin axis of the Earth provides the z-direction, the x-axis is on the equatorial plane, aligned with Greenwich meridian (i.e. *Longitude* $= 0°$ and the y-axis is $90°$ to the East, to form a right handed reference frame. Geographic coordinates and ground station coordinates are often expressed

[1]The plane of the mean Earth orbit around the Sun

[2]Around March 21^{st}

in ECEF because of the convenience of relating coordinates to fixed points on the Earth. In the ECEF frame the axes rotate with the Earth, so it is not an inertial frame, and if one wants to convert between ECI and ECEF a date and time are needed to compute the hour angle that defines how much the Earth has rotated with respect to the fixed *Vernal Equinox* line. Such angle is commonly called Greenwich Average Sidereal Time (GAST) and it is denoted in this dissertation by the symbol $\delta(t) = \delta_0$. The relation between ECI and ECEF frames is depicted in Fig. 8.

The shape of the Earth is approximated better by an ellipsoid than by a sphere. For this reason the geoid shape has been defined to help identify precisely coordinates of points on the Earth. The geographical coordinates of targets are usually given in geodetic coordinates: i.e. geodetic latitude, longitude, and altitude above the reference geoid. ECEF coordinates are often also called geocentric coordinates, and a transformation from ECEF cartesian coordinates to spherical coordinates gives the (usually called) geocentric latitude, longitude and altitude. The relation between the orbital elements, and thus the orbital frame, and the ECI frame is depicted in Fig. 9.

C. Flower Constellation Theory

Flower Constellations have been introduced while designing Earth-resonant orbits: i.e. orbits for which the sub-satellite point traces a closed path on the earth surface[3]. More in general, a key property to determine the parameters of a Flower Constellation is that of being synchronized with some arbitrary rotating frame. In the following, most of the discussion will be done in the ECEF frame, without any loss of generality since most planets in the Solar System spin near uniformly around an analogous

[3]Or, equivalently, the radius vector traces a closed path in the ECEF coordinate frame

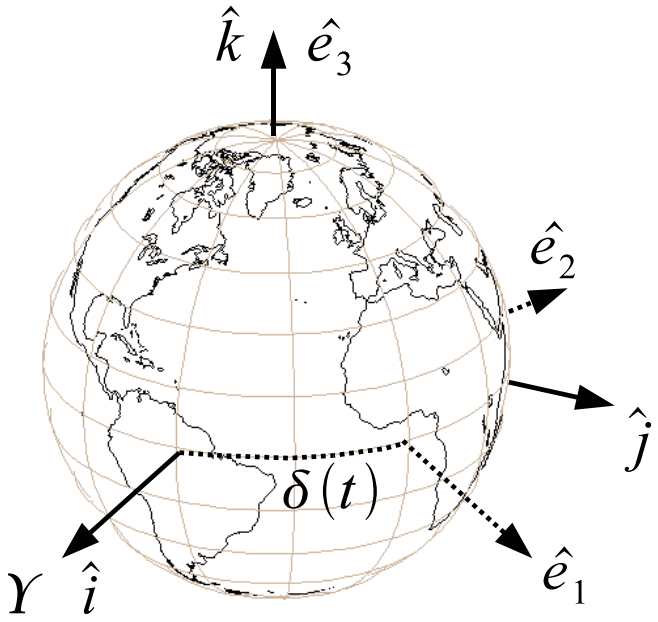


Fig. 8. Relation between ECI $\{\hat{i}, \hat{j}, \hat{k}\}$ frame and ECEF $\{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$; the angle between $\hat{i}$ and $\hat{e}_1$, $\delta(t)$, is the GAST angle and it is a function of time since the ECEF frame spins with the Earth.

"North Pole" axis.

If a closed trajectory is desired in a rotating reference frame, such ECEF, eq. (3.1) must be satisfied:

$$N_p\, T = N_d\, T_\oplus \tag{3.1}$$

in which T is the orbital period, $T_\oplus$ is the *Earth Sidereal Period*, whereas N_p and N_d are two arbitrary integers; resonant orbits are also called *compatible orbits* by some authors. Eq. (3.1) simply states that the spacecraft completes N_p orbits in N_d days. The parameters N_p and N_d therefore define the resonant orbit period. The fact that the constellation is resonant with the Earth spin period is a design choice that endows the Flower Constellations with the desirable property of having a compatible orbits[4],

[4]Sometimes also called resonant orbits or repeating ground track

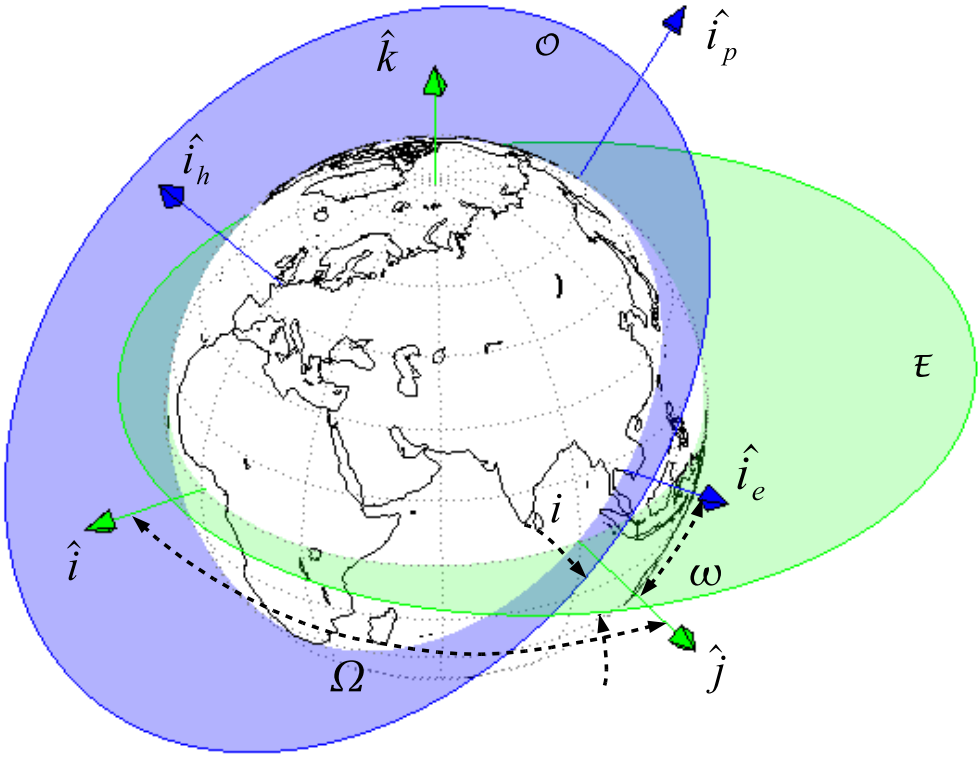


Fig. 9. Relations between the ECI frame $\{\hat{i}, \hat{j}, \hat{k}\}$ and the Orbital frame $\mathcal{O} = \{\hat{i}_e, \hat{i}_p, \hat{i}_h\}$; The orbit is rotated from the reference position E to the actual orbital plane, $\mathcal{O}$, through a $3-1-3$ euler rotation sequence, with the angles $\Omega = 90°$, $i = 30°$, $\omega = 15°$.

but it should in no way be seen as a limitation of the method: one is completely free to synchronize the constellation with any arbitrary spinning reference frame.

The parameter N_p, being the number of loops the spacecraft completes before closing the ECEF trajectory, determines how many *petals* the trajectory has; in fact, it is after this characteristic shape of the closed trajectory in the ECEF frame that Flower Constellations are named, as it can be seen in Fig. 10(a). Fig. 10(b) shows that the petal shape of the relative path in the ECEF coordinates is not a constant trait of

Flower Constellations, but largely depends by the choice of the argument of perigee. From eq. (3.1) it can also immediately be seen that if N_p and N_d have common factors the resulting orbital period is identical and therefore the non restrictive hypothesis that N_p and N_d are chosen to be coprime (i.e. without common factors) is also assumed.

The number of satellites that constitute the constellation is encoded in another integer, N_s. Then, as already seen for Walker constellations, one must decide how the satellites are distributed in how many orbital planes: this issue, referred to as *Phasing*, is explained in details in [35], and it is summarized here for completeness.

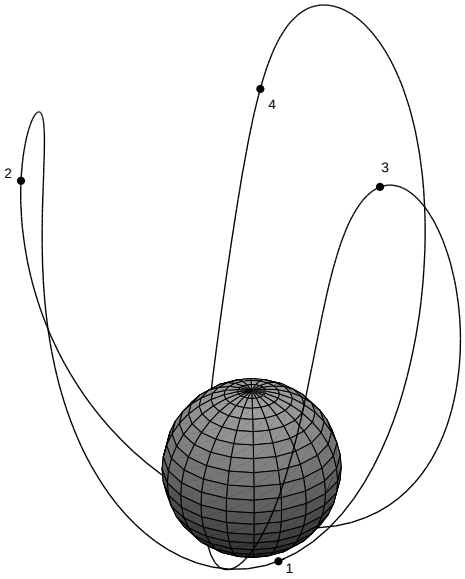
$$\Omega_{k+1} = \left(\Omega_k + 2\pi \frac{F_n}{F_d}\right) \mod 2\pi \tag{3.2a}$$

$$M_{k+1} = \left(M_k - 2\pi \frac{F_n N_p + F_d F_h}{F_d N_d}\right) \mod 2\pi \tag{3.2b}$$

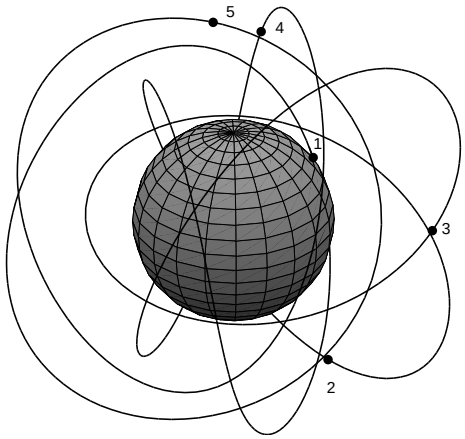
Equations (3.2) define how many orbital planes are being used and how the satellites are distributed in each orbital plane, so that they all follow the same trajectory in the ECEF coordinate frame[5]. In particular eq. (3.2a) defines how the orbital planes are distributed around the central body: e.g. if $F_n = 1$ and $F_d = 3$ it becomes clear that 3 orbital planes are distributed 120° apart, i.e. symmetrically, around the Earth. Thus F_d is the number of orbital planes and F_n defines the spacing around the central body. Given that in eq. (3.2a) there is a ratio of natural numbers, all configurations in which F_n, and F_d yield the same ratio will have the same orbital planes distribution.

Eq. (3.2b) defines the initial (i.e. at $t = 0$) *Mean Anomaly* of the satellites so that they will all follow the same relative trajectory. For this to happen satellites can not be simply placed anywhere on their orbit, but their placement must follow a

[5]This trajectory will be called simply *relative trajectory* in the following, since it is relative to the fixed Earth.



(a) Broglio FC



(b) Example FC

Fig. 10. (a) The first discovered Flower Constellation. This constellation has $N_p = 3, N_d = 1, N_s = 4$ and it is critically inclined, $i = 63.4°$. The petals are uniformly distributed at 120° thus this configuration allows for simultaneous observation of Europe, Asia and North America. Example (b) is an FC with 5 satellites that remain always on the same side, within 90° of longitude, while following the ECEF relative trajectory; the trajectory loses the petal shape because of the particular choice of $\omega = 0$.

rule: the position of satellite $k+1$ must be such that an observer on the Earth sees satellite $k+1$ at time t_0 as it would see satellite k at time $t_0 + \Delta t$ with Δt being the time required for Earth to spin by an angle equal to the difference in RAAN between satellite plane Ω_k and Ω_{k+1}. The number of available slots is, therefore, limited and, sometimes, one must adjust the phasing parameters so that the desired number of satellites can be placed on the chosen relative path. This rule, introduced in [29] actually gives the phasing eq. (3.3): the parameter $F_h \in \{0, \ldots, Fd-1\}$ has been introduced later in [36] to control the order in which the satellites are placed in the available slots, but does not modify the fundamental idea behind the rule.

$$M_k = M_k - 2\pi \frac{F_n N_p}{F_d N_d} \tag{3.3}$$

According to eq. (3.2b) the upper limit to the number of available slots per orbit is given by the parameter N_d, but the actual maximum number of satellites per orbit depends on all the integer parameters since the *modulo* operation in eq. (3.2) may cause the repetition of pairs $< \Omega, M >$ and thus the actual number of satellites per orbit allowed by the distribution could be smaller than N_d. It follows that the number of satellites for a FC is bounded by: $N_{s,\max} \leq F_d N_d$. An algorithm to compute the actual maximum number of satellite per orbit for a given FC is given in alg. (3) which is a summary of the results given in [35].

As mentioned before some configurations do not fill all the $N_d F_d$ slots: when this happens a *secondary path* [37] is formed. A *secondary path* is so called because the satellites motion as a whole seem to be following a pattern that is not expected by looking at the motion of the satellites in either ECI or ECEF frame. Such "emergent behavior" patterns are very difficult to explain with words or even with a static image and the best way to understand this issue is by utilizing the FCVAT program.

Finally, in order to highlight differences and similarity with the classical con-

Algorithm 3 NumSatMax$(N_p, N_d, F_n, F_d, F_h)$

Require: $N_p \in \mathbb{N}$, number of petals, $N_d \in \mathbb{N}$, number of days, $F_n \in \mathbb{N}$, phase numerator, $F_d \in \mathbb{N}$, phase denominator, and $F_h \in \mathbb{N}$ step parameter.

Ensure: $N_{s,\max}$ maximum number of satellite slots in the constellation

1: $g \leftarrow \gcd(F_n N_p + F_d F_h,\ F_d N_d)$
2: $R_d \leftarrow \dfrac{F_d N_d}{g}$
3: $C_r \leftarrow \gcd(R_d, F_d)$
4: $N_{s,\max} \leftarrow \dfrac{R_d}{C_r} F_d$

stellation design method, Table I compares capabilities of Flower Constellations and Walker Constellations. Table I introduces a few new definitions, briefly: *Two way orbits* are defined in [38] as:

> A set of compatible orbits whose ground track path is a closed-loop trajectory that intersects itself, in some points, with tangent intersections. The spacecraft passes over these tangent intersections once in a prograde and once in a retrograde mode.

In practice, it is possible for a satellite to revisit the same site once from apogee and once from perigee, enabling observations of the same target at different resolutions. *Dual Compatible Orbits* are designed to be resonant simultaneously with two rotating reference frames (e.g. ECEF and mean motion of another satellite orbit, see [39]).

Finally, it must be mentioned that when using non-critically inclined elliptical orbits the effect of perturbations, $\overline{\dot{\omega}}$ will cause the perigee to precede, therefore the control effort for station keeping must be taken into consideration for the fuel budget.

Inducing slight variations away from critical inclination could be used to reconfigure a FC, if desired, and then restore the critical inclination upon completion of

Table I. Comparison of capabilities of Flower and Walker Constellation design methodologies.

Property	*Flower*	*Walker*
Circular Orbits	Yes	Yes
Elliptical Orbits	Yes	No
Free choice of inclination	Yes	Yes
Choice of revisiting time with multiple satellites	Yes	No
Compatible orbits If desired If desired	Yes	No
Two way orbits	Yes	No
Dual Compatible orbits	Yes	No
Multi-stationary Orbits	Yes	No
Sun Synchronous	Yes	Yes
Arbitrary number of satellites	Yes	No
Secondary Path	Yes	No

the reconfiguration.

D. Compatible Orbits

If the linear J_2 effect is considered, then eq. 3.1 must be modified to take into account node line precession and the effect of perturbations on the orbital period. For this reason the angular velocity of the Earth is compounded with the node line precession rate and thus the ECEF frame rotates with an apparent angular velocity $\tilde{\omega}_\oplus = w_\oplus - \overline{\dot{\Omega}}$, thus the *Nodal Period of Greenwich* is defined as: $T_{\Omega G} = 2\pi/\tilde{\omega}_\oplus$. Similarly the spacecraft nodal period, i.e. the period in which a spacecraft crosses the node line, is computed as: $T_\Omega = 2\pi/(\overline{\dot{\omega}} + \overline{\dot{M}})$. Thus, taking into account the J_2

linear perturbations, a compatible (or resonant) orbit satisfies eq. (3.4).

$$N_p\, T_\Omega = N_d\, T_{\Omega G} \tag{3.4}$$

which becomes:

$$N_p\,(w_\oplus - \overline{\dot{\Omega}}) = N_d\,(\overline{\dot{\omega}} + \overline{\dot{M}}) \tag{3.5}$$

After assigning values for N_d, N_p, e, i, we must solve eq. (3.5) in order to find the correct value of the orbit semi-major axis required to obtain a compatible orbit. Equations (2.37f), (2.37d), and (2.37e) can be re-written in the following compact forms, respectively:

$$\overline{\dot{M}} = \sqrt{\frac{\mu}{a^3}}\,\left(1 + \frac{c_m}{a^2}\right) \tag{3.6a}$$

$$\overline{\dot{\Omega}} = c_\Omega\,\frac{a^2 + c_m}{a^4}\,\sqrt{\frac{\mu}{a^3}} \tag{3.6b}$$

$$\overline{\dot{\omega}} = c_\omega\,\frac{a^2 + c_m}{a^4}\,\sqrt{\frac{\mu}{a^3}} \tag{3.6c}$$

where the following constants have been defined:

$$c_m = \frac{3}{4}\,J_2\,\left(\frac{R_\oplus}{1-e^2}\right)^2\,(2 - 3\sin^2 i)\,\sqrt{1-e^2} \tag{3.7}$$

$$c_\Omega = -\frac{3}{2}\,J_2\,\left(\frac{R_\oplus}{1-e^2}\right)^2\,\cos i \tag{3.8}$$

$$c_\omega = \frac{3}{4}\,J_2\,\left(\frac{R_\oplus}{1-e^2}\right)^2\,(5\cos^2 i - 1) \tag{3.9}$$

Substituting eq.'s (3.6a), (3.6b), and (3.6c) in eq. (3.5), and setting:

$$c_n = N_p\, c_\Omega + N_d\, c_\omega \tag{3.10}$$

the following function is obtained:

$$f(a) = N_p\, w_\oplus - \sqrt{\frac{\mu}{a^3}}\,\left(N_d + \frac{c_n}{a^2}\right)\left(1 + \frac{c_m}{a^2}\right) = 0 \tag{3.11}$$

eq. (3.11) can be solved using Newton-Raphson iterative method, i.e.:

$$a_{k+1} = a_k - \frac{f(a_k)}{f'(a_k)} \tag{3.12}$$

where, by setting $c_0 = c_m\, N_d + c_n$, the derivative has the expression:

$$f'(a) = \frac{n}{2\,a}\,\left(3\,N_d + 7\,\frac{c_0}{a^2} + 11\,\frac{c_m c_n}{a^4}\right) \tag{3.13}$$

and where the starting point a_0 can be chosen as the semi-major axis of a compatible un-perturbed orbit:

$$a_0 = \mu\left(\frac{N_d}{N_p\, w_\oplus}\right)^{2/3} \tag{3.14}$$

Wagner in [40] presented a similar iterative method to calculate the mean semi-major axis required for a compatible orbit. The algorithm begins with the evaluation, as a starting point, of the unperturbed semi-major axis a_0 given in eq. (3.14). Then the semi-major axis is evaluated through an iterative process using the following iterative equation:

$$a_{k+1} = a_0\,\left[1 + \frac{3}{4}\,J_2\,\left(\frac{R_\oplus}{a_k}\right)^2\,(2 - 3\sin^2 i)\right]^{2/3} \cdot \left[1 - \frac{3}{4}\,J_2\,\left(\frac{R_\oplus}{a_k}\right)^2\,\left(1 + \frac{2\,N_p}{N_d}\,\cos i - 5\cos^2 i\right)\right]^{2/3} \tag{3.15}$$

The procedure can be stopped when the desired accuracy has been reached, and usually does not require more than two or three iterations to achieve order of *cm* precision.

CHAPTER IV

OPTIMIZATION TECHNIQUES

A review of all the optimization techniques is far beyond the scope of this chapter and therefore only the main points will be summarized in order to provide context and motivations for the design choices exposed in the implementation chapter. The interested reader can follow the many references to the existing literature.

Global optimization is the process of finding a maximum of a selected function $f(\mathbf{x})$ over the whole domain of the function. In mathematical terms:

Definition. (*Global minimum*) Given a function $f : S \to \mathbb{R}$, $f^* = f(\mathbf{x}^*) > -\infty$ is called a global minimum if and only if

$$\forall \mathbf{x} \in S : f^* \leqslant f(\mathbf{x}) \tag{4.1}$$

Finding a minimum or a maximum is essentially the same thing (one can always change the sign of a cost function) and therefore only minimization problems are considered in the following. Optimization methods can be classified in few broad categories:

- Analytical methods (or indirect search).
- Direct search.
- Enumeration.
- Random search.

Analytical methods seek extrema of a cost function by computing its gradient and setting it to 0. The resulting set of non-linear equations is then solved to yield the extrema. Direct search methods instead are based on the idea of *hill climbing*.

Since the gradient of a function points toward the direction of maximum increase, by going in the opposite direction one goes in the direction of maximum descent and by iterating this process local minimization is achieved [41].

Direct and indirect methods have been studied extensively and the literature on the subject is massive, see [42] for example. They are ubiquitous and their strongest appeal is the solid mathematical foundations upon which they have been developed. Even with all these virtues however they still present important weaknesses: analytical methods require an analytic cost function and the computation and solution of a set of non-linear equations, which may not always be practical. They also require the cost function to be differentiable.

Direct search methods instead require a starting point close to the minimum and convergence is not guaranteed. Furthermore, both methods are local in nature. Strictly speaking direct search methods do not require the analytical computation of the gradient, it can be approximated numerically, but they still require the cost function to be smooth, which is not always the case for many important problems.

Enumerative schemes may be very appealing when the search space is small. In this case the cost function is evaluated for all the points in the search space and the best solution is selected. If the search space is continuous, then it can be discretized to allow a finite number of choices. Enumeration is interesting because it is very simple to implement, provides global minima, can be easily parallelized, and becomes ever more attractive for even moderately large problems as computers and memory become cheaper and processors always more powerful; it is however fundamentally inefficient and many practical problem do have very large or even infinite search spaces. Dynamic programming also falls into the domain of enumerative schemes and although it employs much finer algorithms than naive enumeration, still suffers from the so called Bellman "Curse of Dimensionality", rendering it inefficient for problems

of moderate to large size.

The term *Random search* may sound odd: it really means "minimization by using a stochastic approach". In this context *random* does not mean "without direction", but rather that *random choice* is utilized to guide the search process. The meaning of the latter sentence will become more clear by reading the following digression about Genetic Algorithms which is the random search method chosen for part of the constellation optimization work in the remainder of this dissertation.

A. Genetic Algorithms

Evolutionary Algorithms are meta-heuristic, global optimization algorithms that model with computer code the mechanisms of biological evolution in a population of individuals (candidate solutions in our context) [43]. This generic term covers those algorithms that use genetic operators such as recombination, reproduction and mutation, guided by the general principle of *survival of the fittest*.

Candidate solutions to the optimization problem play the role of individuals in a population and their *fitness* is measured by a user defined cost function. The fittest members are then selected for reproduction, recombined and then mutated to build a new generation of candidate solutions. Through successive iterations the candidate solutions evolve towards a set of solutions whose average fitness is higher than that of the original population. EAs differ from random search: they employ randomization as a tool to guide the search, while random search is expected, on average, to not perform any better than enumeration.

Enumeration, for many practical parameter spaces, is simply too computationally expensive and slow. It should be mentioned that the so called "No Free Lunch Theorem" [44] implies that EAs do not perform better than random search if the

performance is averaged on all the possible cost functions. However, for practical problems and by exploiting the knowledge of the problem to tune the search parameters, EAs have been successfully applied to diverse domains and optimization problems. There are four major categories of EAs: Genetic Algorithms (GAs), Genetic Programming (GP), Evolution Strategy (ES), and Evolutionary Programming (EP). The last two are similar in the mechanics but have been developed separately.

GAs are by far the most common implementation of EA. In GAs, individuals, i.e. candidate solutions, take the form of string of bits that encode generic properties (genes) of the desired solution. An initial population of individuals is randomly generated and the genetic operators (selection, recombination, mutation) are then applied in order to evolve the population. The iteration stops when the relative increment of the fittest member with respect to the previous iteration reaches a predefined tolerance or when a maximum number of iteration has been completed. GAs have also been the preferred choice to solve optimization problems in which the cost function is governed by both integer and real parameters; in these cases classical gradient based techniques can not be used because the function is not continuous. Even when the function is differentiable, gradient methods require a starting point "close" to the optimal solution and convergence is not guaranteed. Thus a GA could be used to find a good initial condition, and then a gradient method applied to refine the best solution found by the GA in order to ensure true optimality. Such hybrid schemes, where possible, have been used since the infancy of GAs.

There are alternatives to GAs for global optimization; two noteworthy ones are: Simulated Annealing and Particle Swarms. Simulate Annealing [45] uses stochastic processes to guide the search towards states of minimum energy, by simulating a process used in metallurgy involving heating and cooling of alloys to reduce defects by increasing the size of crystals in the metal. Particle Swarms is probably the newest

global optimization technique introduced in [46]. It uses the concept of a swarm of particles interconnected in neighborhoods through a sort of social network (i.e. the ability to communicate). Each particle state is made of: its current position, its velocity, its own fitness, and the best solution found in its neighborhood. Movement through the search space is guided by the interaction of the particles in their neighborhood and to the spreading of "good news", i.e. the fact that a good solution has been found by some particle.

There is no conclusive evidence that any of these meta-heuristic global optimization methods (Particle Swarms, Simulated Annealing, Evolutionary Algorithms) performs definitely better than the others: they all require fine tuning of the algorithm parameters for the specific problem at hand and there are no universally accepted methodologies to accomplish this tuning which has been likened more to an art rather than to a well defined procedure. GAs however have a practical advantage over these competing techniques: stable, robust implementation are freely available in C++ and a GA Toolbox is also available for MATLAB [47]. The latter then is the preferred choice in this work since the algorithm provided with MATLAB is general, flexible, and easily interfaced with the rest of the MATLAB code.

As it should become clear from the algorithm depicted in Fig. 11 the meaning of *Optimization* in the context of GA should be carefully understood to avoid confusion. GAs in fact do not guarantee that when the algorithm reaches a stopping condition the fittest individual, i.e. the best solution examined, corresponds to the global maximum of the cost function in a mathematical sense, i.e. the best possible solution. A correct statement would be that the algorithm, having explored a number of solutions and having improved the average fitness of the population from one generation to the next, is providing a sub-optimal solution, corresponding to the fittest individual examined during the evolution process. This behavior is the source of many discussions and

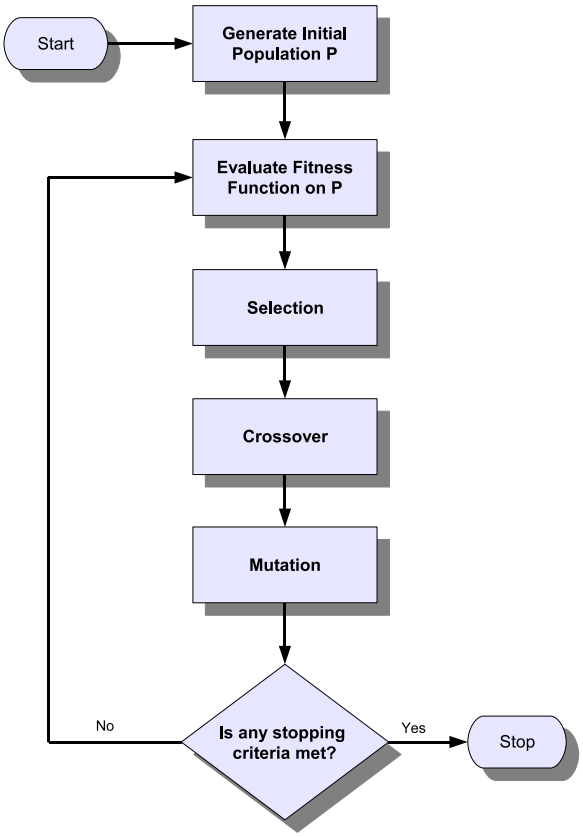


Fig. 11. Flow chart for a typical GA implementation.

polarization of opinions on the validity of GAs.

However it is the writer's opinion that this fact should not be viewed as a reason to reject GAs: they have been successfully utilized in many practical applications for decades. In many real world problems even the definition of what the optimality criteria are can be elusive: several competing factors must be taken into account when deciding what the "best" solution means[1]; the knowledge of all the factors influencing the problem may be incomplete and other factors that can not be quantitatively assessed may influence the choice of the best solution[2]; in such cases, as humans, we

[1]Multi objective optimization algorithms may be utilized to address these kind of problems.

[2]E.g. political issues, strategic or social concerns.

are accustomed to dealing with sub-optimal solutions constantly.

In the GA literature some terms commonly used in biology have been imported for analogy. The input vector to the *fitness function* (i.e. objective function) is called *chromosome*. Each variable in the input vector is called a *gene*[3]. The set of all candidate solutions is called population and each iteration is called *generation*; the optimization process itself is sometimes named *evolution*.

Genetic Operators

The flowchart in Fig. 11 depicts the classical implementation of a GA, but the operators can be implemented in many different ways and the results may vary accordingly. The algorithm starts with generating a random *initial population* (a) of predefined size. In the following phase, *selection* (b), the individuals of the population are matched by a stochastic selection algorithm according to their fitness (the best individuals having higher chance of being selected) and their genes are recombined by the *crossover* operator into new individuals. The mutation operator is applied randomly to the new offsprings (c) and the fitness function is evaluated for the new population. Finally (d) the stopping criteria are evaluated and, if not met another iteration is started or otherwise the algorithm stops producing its output: the final population, the best individual, and some statistics for a posteriori analysis of the algorithm behavior (useful for debugging and tuning).

The encoding of the input variables in the chromosome can be done either by a string of bits or by a vector of real numbers or even by a list of variables from heterogeneous domains. In the classical GA implementation the chromosome is encoded with a string of bits and each variable (*gene*) is represented by a sub-string of bits. If

[3]When the *genes* assume specific values in an individual, they are called *alleles*.

the domain of the variable is a subset of the real numbers, i.e. an interval $[a, b] \in \mathbb{R}$, the precision of representation is determined by the number of quantization bits (n): the accuracy obtainable with n bits is $(b - a)/(2^n - 1)$. The representation with string of bits is in fact the most general, since virtually any information can be represented easily with an appropriate number of bits. The uniformity of representation of all the encoded variables makes the implementation of the genetic operators very easy. An encoding and decoding step of the chromosome variables must be added in this case for the evaluation of the cost function. In order to avoid encoding and decoding vectors of real numbers can be chosen too to represent chromosomes: it is sometimes preferred for being a more direct representation of the input vector to the cost function; it does have however the problem that such representation makes the implementation of the operators more complicated, as it should become clear in the following.

The genetic operators, *selection*, *crossover*, *mutation*, can be implemented in many different ways; the classical implementation, with some commonly used variations, is now described briefly.

The *selection* operator determines which pairs of individuals are selected to generate offsprings (i.e. new solutions). Parents could be selected completely at random but most GA implementations tend to bias the selection towards the best individuals (i.e. *Selection Pressure*). A popular selection scheme is the *roulette wheel* scheme[4]. In the *roulette wheel* algorithm, represented graphically in Fig. 12, each individual is assigned a sector of a virtual wheel that is proportional to the individual own fitness; the area of the sector is the probability that an individual will be selected: $Pr\{\text{i selected}\} = f_i / \sum f_i$. This scheme, as for most selection algorithms, still allows

[4]Also called *stochastic sampling with replacement*

weaker individuals to generate offsprings. This fact is important to prevent premature convergence, a phenomenon that tends to narrow the search space close to the best individuals, thus increasing the risk of finding local minima.

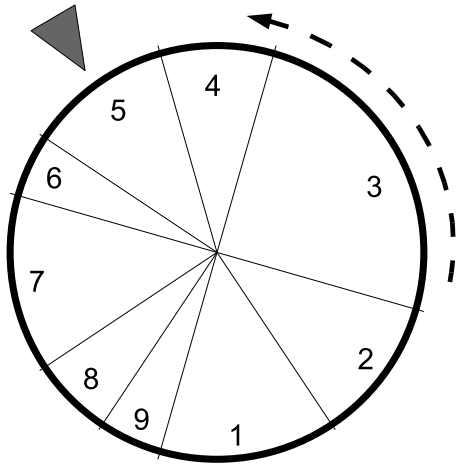


Fig. 12. *Roulette wheel* selection graphical representation for a population of 9 individuals. The slice of each individual is proportional to the individual fitness. The wheel is spun and the individual pointed to by the arrow on the upper left corner is selected.

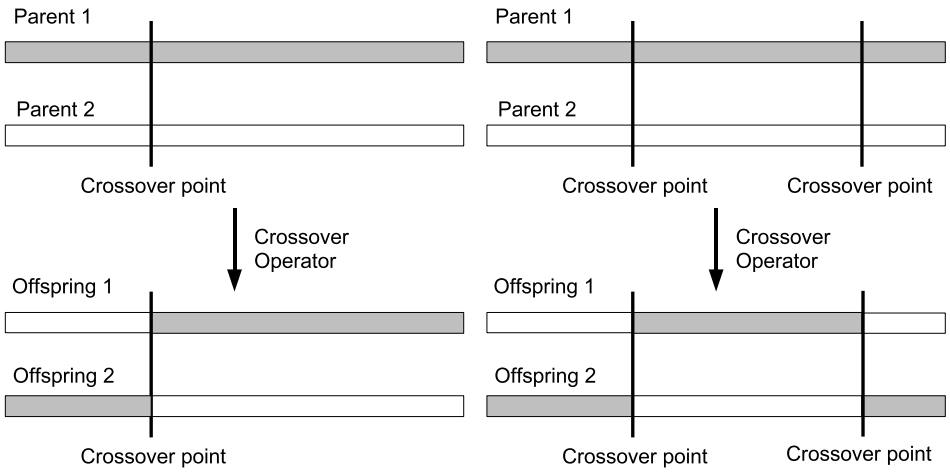
A good selection algorithm should be easy to implement and should allow for tunable selection pressure. The selection pressure should be tuned keeping in mind that a higher selection pressure (i.e. best individuals are highly favored) tends to increase the chance of finding sub-optimal solutions (local minima), whereas a lower selection pressure reduces the convergence rate (no good solutions are found). For this reason another popular scheme for selection is the *tournament selection* [48]. With this scheme a tournament (of size k) is held between k individuals chosen at random in the population. The tournament winner (i.e. individual with highest fitness) is inserted in a mating pool. The mating pool tend to have an average fitness higher than the average fitness of the current generation: this determines a selection

pressure. Tuning of the desired level of selection pressure in a particular application domain is achieved by changing the size of the tournament: the winner of bigger tournaments tends to be an individual with higher fitness than the average winner of small tournaments [5].

Crossover is the genetic operator that generates new offsprings from the chromosomes of two parent individuals through recombination of their genome. In this case too there are different choices for the implementation. The basic scheme is single point crossover; each of the two parent chromosomes $A = \{x_1, \ldots, x_n\}$, and $B = \{y_1, \ldots, y_n\}$ is split at some random point $i = 1 \ldots n$ and the generated offspring will have chromosomes $C = \{x_1, \ldots, x_i, y_{i+1}, \ldots, y_n\}$ and $D = \{y_1, \ldots, y_i, x_{i+1}, \ldots, x_n\}$ in which x_i and y_i are variable representing values in $\{0, 1\}$ or in $\mathbb{R}$, according to the chosen representation. The crossover operator remains basically the same whether the chromosome is an array of real variables or a string of bits. This simple scheme has been adjusted and rehashed in many flavors; a graphic representation of the operator is in Fig. 13. Multi-point crossover, i.e. crossover in which there are multiple split points, is a relatively popular choice and even the more radical *Uniform Crossover*, i.e. homologous bits in the parents chromosomes are swapped with fixed probability p, has been utilized successfully. Some other form of Evolutionary Algorithms, such Evolutionary Programming (EP) [49] disregard *Crossover*, i.e. the mechanism of recombination, entirely and use mutation more aggressively instead. EP applies evolution to finite state machines to solve simple problems. Goldberg criticizes this approach for ignoring the important mechanism of recombination, and suggests that this is the cause for the inability of solving more than just small problems with EP.

Finally the *mutation* operator randomly changes the value of genes in the chro-

[5] As the sport fans know well



(a) Single Point Crossover (b) Multi point crossover

Fig. 13. Pictorial representation of the crossover operator.

mosome with some, typically very small, probability. If the chromosome is implemented as a string of bits, mutation is as simple as flipping the value of some bits of the chromosome, whereas if the chromosome is implemented as a real array the value of a variable in the array is modified by adding a Gaussian perturbation of chosen variance. The mutation operator has the effect of extending the search beyond the specific portion of the search space spanned by recombination alone and therefore helps guard against premature convergence. The usefulness of mutation can be better understood with a simple, if extreme, example. Suppose that the initial population is chosen at random and that for some twist of fate all the chromosomes are initialized to strings of only zeros. Clearly the recombination mechanism alone could not produce anything but more strings of zeros. In this peculiar example mutation becomes essential to allow the GA to move away from the region in which it would be stuck by such a peculiar choice of the initial population. On the other hand if the probability of mutation is set too high, the behavior of the GA becomes proportionally more similar to a random search. This fact is not surprising: it has been suggested

regarding the evolution of life on Earth that it is the mutation of genetic material caused by high energy particles that accounts for the bio-diversity on our planet, and that if the quantity of high frequency radiation reaching the Earth had been higher or lower than it is, there might be no life at all.

B. Strength and Issues of Genetic Algorithms

Goldberg in [50] describes Holland's hypothesis of *schemata*, a very interesting insight on why Genetic Algorithms work; he argues that the preservation of basic building blocks (*schemata*, more technically) is the fundamental mechanism that makes GAs successful. If one imagine a *Chromosome* as a string of bits, then *schemata* are short sub-sequences of consecutive bits. Goldberg points out that the crossover and mutation operators tend to preserve short schemata in highly fit solutions and since in each generation order of n^3 schemata are processed as opposed to order of n function evaluations[6] there is an *implicit parallelism*, apparently unique to GAs, that allows a fast reproduction of building blocks from good solutions within the population. This hypothesis, although convincing, has not been rigorously proved to date, and some studies seem to indicate that it may not be correct, or at least not the "whole story." In particular Syswerda [51] and Fogel [52] criticizes the hypothesis by showing that uniform and multi-point crossover, which are highly disruptive for schemata, seem to outperform single point crossover, at least for the application considered in their works.

Notwithstanding these well founded criticisms, GAs have been used in many domains with great success because of their inherent virtues:

- GA implementation is simple and does not depend on the domain of application;

[6] n is the population size.

- GAs can be used for global optimization and perform well even in large, multi-dimensional search spaces;
- Convergence is not adversely affected by initial conditions far from the optimum;
- Any cost function is acceptable: differentiability is not required;
- They mimic a widespread and well known natural process: evolution.

As for all good things, even for the virtues of GAs there is a price to pay: GAs tend to be computationally expensive since there are many evaluations of the cost function, i.e. order of (*Population Size*) $\times$ (*Maximum Number of Iterations*); moreover, in order to perform well in some specific domain, they require fine tuning. This tuning process is cause for concern to some who say that by tuning the parameters one is simply influencing the algorithm behavior according to one's expectations of what a good solution should be[7].

Clearly a good GA implementation must address these concerns: the complexity concern must be addressed by efficient coding and the choice of cost functions that can be evaluated quickly. The complexity issue can also be addressed by relying on the ever improving computational power of computers, which is still following Moore's empirical law (i.e. doubling each year), and the increasing use of parallel computing for which GAs are very well suited.

As for the second concern, tuning a GA is in fact a delicate art, and one of the criteria used to judge the quality of the chosen cost function and parameters is indeed that the GA should find solutions that satisfy the requirements and can not be easily obtained by other means.

[7]Private conversation with Dr. Landis Markley, Cinqueterre, Italy, August 2003

Handling Constraints

With calculus based methods the enforcement of constraints on the cost function input variables can be done analytically, trough Lagrange Multipliers. A constraint is a function of the input variables of the form: $f(x_1, \ldots, x_n) = 0$. Thus a generic minimization problem could be of the form:

$$\min_{x_1,\ldots,x_n} J(x_1, \ldots, x_n)$$

$$\text{subject to:}$$

$$f_1(x_1, \ldots, x_n) = 0$$

$$f_2(x_1, \ldots, x_n) = 0$$

$$\vdots$$

$$f_p(x_1, \ldots, x_n) = 0$$

$$g_1(x_1, \ldots, x_n) < 0$$

$$g_2(x_1, \ldots, x_n) < 0$$

$$\vdots$$

$$g_q(x_1, \ldots, x_n) < 0$$

where $n, p, q \in \mathbb{Z}$ are the number of variables, the number of equality constraints and the number of inequality constraints respectively.

The question is then, how can GAs handle problems of this form? Whenever a crossover operation, or a mutation, is applied the value of a variable could change in a way that violates some constraint. From what we have seen in this digression there appear to be no mechanism in the GA to prevent this from happening. In fact it is the responsibility of the programmer to implement the genetic operators in such a way that the constraints are not violated.

There are different methods to handle these kind of issues with GA; the first and the simplest, applicable if there are only equality constraints, is to allow the constraint violation and augment the cost function by adding the constraints with an appropriate weight in order to heavily penalize unfeasible solutions:

$$J' = J + w_1 f_1^2 + \ldots + w_2 f_2^2 + \ldots + w_p f_p^2 \tag{4.3}$$

each weight can be adjusted to give more importance to some constraints rather than others; from a practical point of view one could divide the constraints in two classes: *hard* constraints and *soft* constraints. The first class corresponds to "physical impossibility", i.e. the solutions that violate such constraints are plainly unfeasible: in this case the weights corresponding to the constraints should be some very high number that clearly mark the solution as bad. The other class, *soft* constraints, corresponds to cases for which a violation of a constraint means that there is a solution that is less desirable, but not impossible. For example, if one wanted a given orbital transfer to be completed in a given time with a given fuel budget, but the best solution found exceeds the maximum time allocated by a fraction of percent, than we might still consider the solution acceptable.

The criticisms that can be made to such approach are twofold: a) there is no guarantee that the constraints are satisfied, b) it may be inefficient because solutions that are not acceptable may survive several generations and require several unnecessary computations of the cost function. The first issue is another way of rephrasing disbelief regarding GAs: the evolution mechanism will take care of eliminating the unfit individuals[8]. The second issue belongs to the efficiency considerations and it is an intrinsic characteristic of the method: GAs are modeled after nature, and nature

[8] An economist may say "Trust the free market"

is not always efficient, but it is effective.

The inefficiency may become unacceptable if the feasible search space is such that most of the individuals will happen to be unfeasible solutions. In this case the operators, i.e. mutation and crossover, must be implemented in a way that does not generate unfeasible solutions.

Using Bit-Strings to Handle Simple Constraints

A very common type of constraint that can be easily handled by an appropriate coding of the input variables is of the type:

$$x_i \in [a, b] \tag{4.4}$$

where $a, b \in \mathbb{R}$ and $i = 1 \dots n$ is some variable in the chromosome. In this case one may simply quantize the interval $[a, b]$ with the number of bits needed to achieve the desired precision and then encode and decode the variables accordingly. For instance, if the desired precision is ρ, then the number of bins, N, in which the interval must be divided is:

$$N = \left\lceil \frac{b-a}{\rho} \right\rceil \tag{4.5}$$

Thus the number of bits needed to encode such quantization must satisfy:

$$2^{n_b} \geq N + 1 \Rightarrow n_b \geq \lceil \log_2(N+1) \rceil \tag{4.6}$$

Example: $A = [2, 4] \subset \mathbb{R}$, $\rho = 0.01$

$N = 2/0.01 = 200 \Rightarrow n_b \geq \lceil \log_2(201) \rceil \simeq \lceil 7.6511 \rceil = 8$. Thus, at least 8 bits are necessary to quantize the interval with the desired precision, which gives an actual precision that is slightly better than required: $\rho' = 2/255 \simeq 0.0078$.

Now suppose that the string of bits (i.e. *gene*): $X = (01000100)_2 = (68)_{10}$ is

passed to the cost function; the decoding of the actual value x in the interval A is simply:

$$x = \frac{(b-a)}{2^{n_b}-1}X + a \simeq 2.533 \tag{4.7}$$

By using this type of encoding one may easily satisfy the most common type of constraint practically encountered and save efficiency at the same time. The drawback is that accuracy of the solution must be decided beforehand: should it be changed, the decoding/encoding of the chromosome must be changed accordingly. The MATLAB implementation of the GA supports encoding of chromosomes with strings of bits as well as another approach, best suited for real array encoding that enforces constraints for the mutation and crossover operators; quoting from Mathworks on-line help on the GA toolbox [47]:

> The GA solver handles linear constraints and bounds differently from nonlinear constraints. All the linear constraints and bounds are satisfied throughout the optimization. However, GA may not satisfy all the nonlinear constraints at every generation. If GA converges to a solution, the nonlinear constraints will be satisfied at that solution.

In conclusion, within this chapter, the choice of GAs to solve the kind of optimization problems treated in this dissertation has been motivated by comparison with other viable alternatives. Critics of evolutionary algorithms often emphasize the suboptimal character of the solutions obtained by GAs, together with the limitations intrinsic in the way of handling constraints. The short answer to both criticisms is that the mechanism of evolution, an appropriate encoding of the input variables, and an appropriate implementation of the genetic operators take care of addressing these issues. The confidence in the method lies in four decades of successful applications of evolutionary algorithms to the most diverse engineering domains. GAs must not be

seen in competition with calculus based optimization algorithm: the two methods, in fact, complement each other very well, as the success of hybrid GA/Gradient based methods in literature testimony. Finally, the optimization aspect of this dissertation can be easily modified; the key innovations lie elsewhere. Put another way, if the reader prefers to use an alternative optimization process, the remainder of this dissertation can readily accommodate this decision.

CHAPTER V

FCVAT AND FCTOOLBOX DESIGN AND IMPLEMENTATION

When the first ideas about FCs were being formulated, and some prototype computer code was being written with MATLAB, it became increasingly clear that the understanding of FCs would greatly benefit from an application that could provide real time 3D visualization of the configurations under scrutiny. Existing software, like AGI STK, quickly proved unsuited and too cumbersome for the kind of analysis that was necessary. For this reason the work on a tool that streamlined the process of visualizing FCs and that was not encumbered by massive commercial licensing issues was started. The results were the FCVAT application and the FCToolbox for Matlab described in the next sections.

A. Flower Constellation Visualization and Analysis Tool

The FCVAT program, whose main window is shown in Fig. 14, has been written entirely in Java [53], using the Java3D extensions for 3D visualization and the Swing libraries for the Graphical User Interface (GUI). The choice of Java as programming language was favored by:

- **Platform independence**; Java is a semi-interpreted language based on a Virtual Machine (VM) that can be instantiated and integrated in a large variety of Operating Systems (OSs) and embedded devices.
- **Object Oriented design**; Java fully supports the object oriented programming (OOP) paradigm.
- **Abstraction from hardware**; the Java VM provides seamless interfacing with the underlying operating system and device drivers in an abstract, consistent

fashion. Thus there is no need for the programmer to know the details of how the particular file system, graphic card, mouse, etc. work on the specific hardware/OS embodiment.

- **API completeness**; API stands for Application Program Interface: Java provides, as integral part of the standard language, a complete set of libraries (GUI, networking, sound, graphics, cryptography, etc.) that address just about every aspect of modern programming[1].

- **Royalty free**; the Java compiler, Java VM, and the API are freely available from Sun Microsystems, Inc. and other implementors (e.g. IBM) free of charge and royalties, for both educational and commercial use.

- **Eclipse**: simply the best Integrated Development Environment (IDE) available today[2]. Designed specifically for Java, but extensible to other languages too, improves programmer's productivity tremendously. It is freely available for download from the Eclipse consortium. [54].

- **Java3D**: extension API of the Java language to support 3D real time graphics and animation by interfacing and extending OpenGL [55] capabilities.

The price to pay for portability is performance: Java is a semi-interpreted language; although there is a Java compiler, the latter is actually translating the high level language into a binary executable (called *bytecode*) that is executed by the Java VM. Note that the Java VM itself must be executed by the OS in the native language of the host machine, therefore each pair OS/hardware must have its own implementation of the Java VM (and in fact Java is available for practically any platform).

[1]Where it unfortunately is somewhat lacking is numerical computation.

[2]Eclipse itself is written in Java.

For most implementations, one can expect Java to use more memory than C or C++ programs[3]; efficiency in memory management has been somewhat compromised for robustness: there are no pointers in Java. Pointers are very useful but also a constant source of memory leaks and unwanted side effects for C and C++ programmers[4]. For this reason Java use a *Garbage Collector* mechanism to reclaim unused memory freeing the programmer from the need of a careful management of pointers.

All in all performances are adequate for most programming tasks and the independence from hardware and OS makes Java really shine in applications that require this feature. In addition to this, it offers a complete set of foundation libraries most of which are not present in the C++ standard library; this is not meant as a criticism of C++: the lack of foundation libraries in C++ is due to a design choice, not a lack of vision.

If many APIs are included in the standard library of a computer language their implementation might prove inadequate for some class of users. The philosophy of C++ is that specialized libraries are available from multiple sources and the software engineer must choose the ones that are more adequate to the programming task at hand. Fig. 14 shows the appearance of the program main window and Fig. 15 shows the software layers on top of which the FCVAT program is built.

FCVAT is designed to be an interactive simulation application. The main requirements that shaped the software design have been:

1. Portability across different platforms.

[3]Java is compared with C++ because it has been heavily inspired by it, as the syntax testimony.

[4]Some would say "sloppy" programmers.

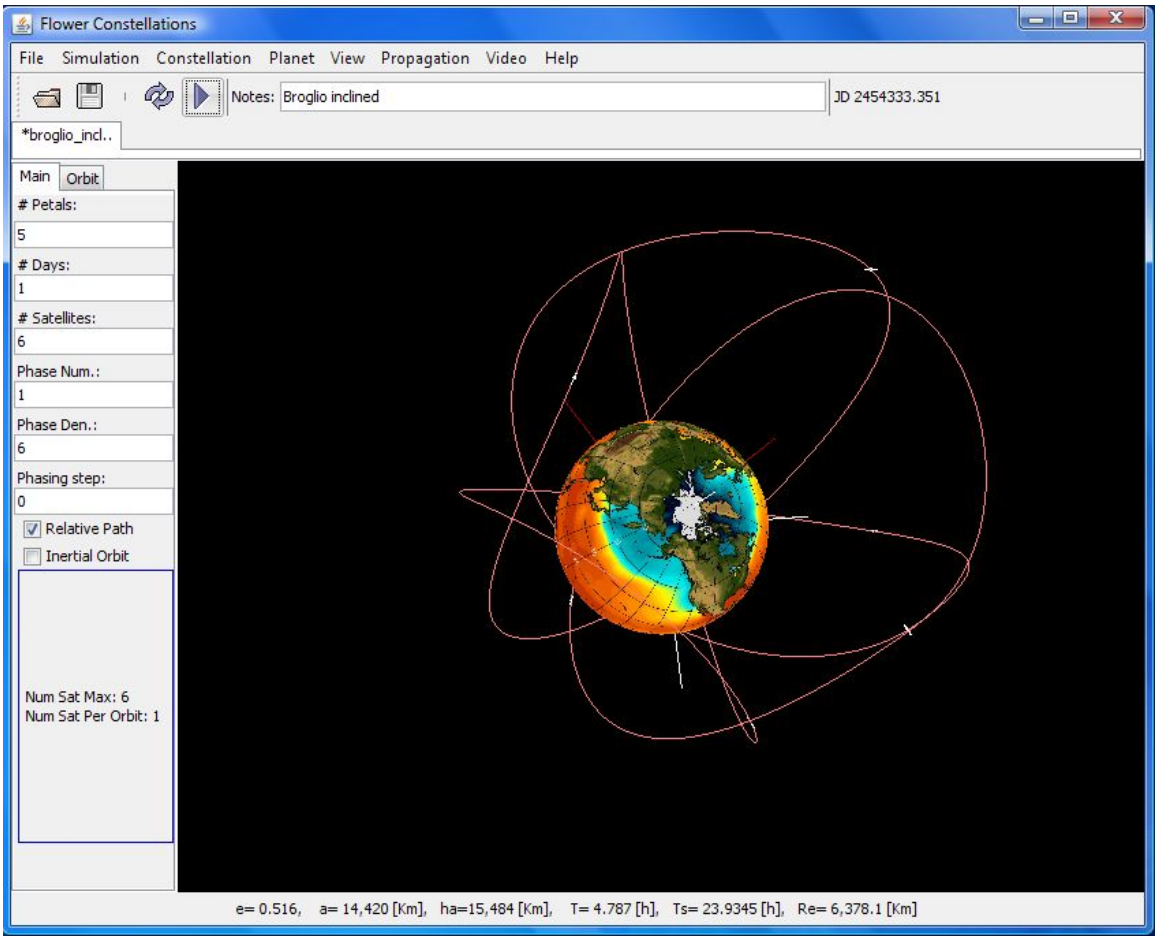


Fig. 14. Main FCVAT window with a $N_p = 5$, $N_d = 1$ constellation animation.

2. No licensing issues and freedom of installation on as many computers as needed.

3. High resolution, real time animation: should be usable for live presentations, seminars, etc.

4. Ease of use: the program must support an "edit and continue" pattern of use; i.e. the user is not required to restart the application to see the effect of some parameter change. Interactive input is therefore required, as it is shown in Fig. 16.

5. Modularity and extensibility: the tool must grow as the visualization and anal-

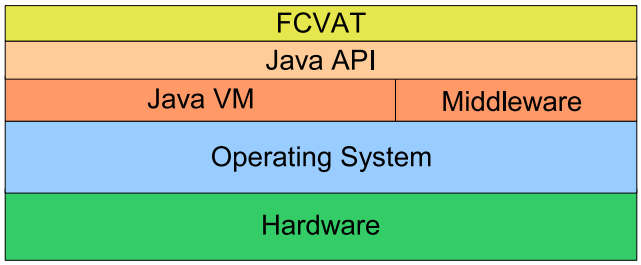


Fig. 15. Software stack graphics representation for FCVAT program.

ysis necessities for FCs grow.

6. Interoperability with other software, like AGI STK (r) and MATLAB (r).

The first two requirements have been satisfied by the choice of the programming language. The third requirement is satisfied by the use of Java3D and Swing libraries. Modularity is achieved through an Object Oriented design and, finally, interoperability is achieved by exporting data in formats readable by other programs.

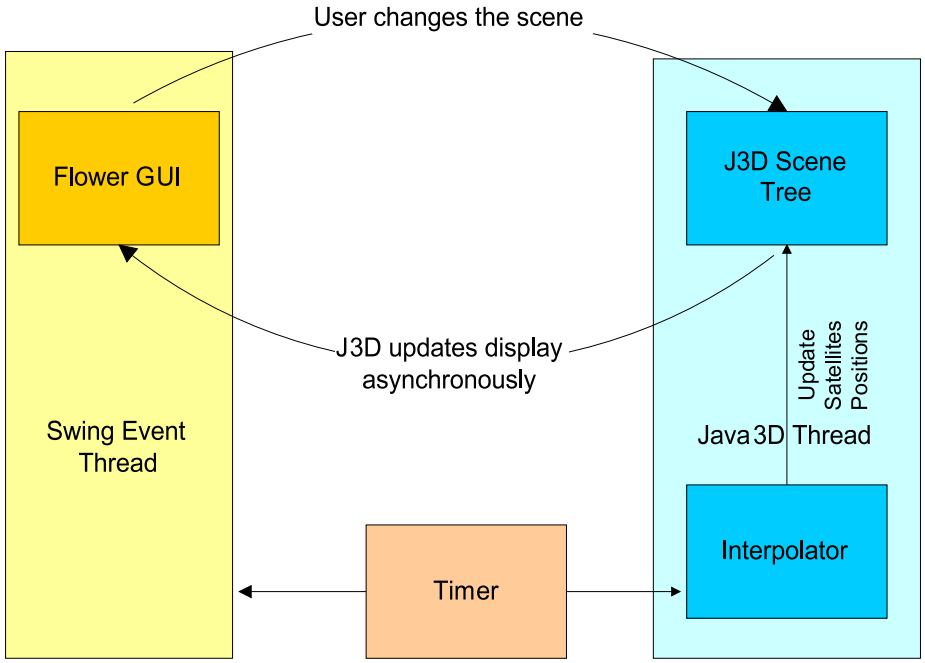


Fig. 16. Edit/Update animation cycle of FCVAT.

The Object Oriented design methodology proposes to model a problem as relations of objects that maintain an internal state (i.e. variables) and expose an interface (i.e. functions) to read and modify such state. A program is viewed as a flow of interactions between *objects*. The Unified Modeling Language (UML) [56] is a well established industry standard to design and describe object oriented software: the diagrams presented in this section have been generated according to the UML specifications. Figs. 17 and 18 describe the main components of the FCVAT application and how they are related. The `Flower` class is the GUI that manages the interaction with all the other objects, in the usual functional programming terminology it is similar to the `main` function of the `C` programming language.

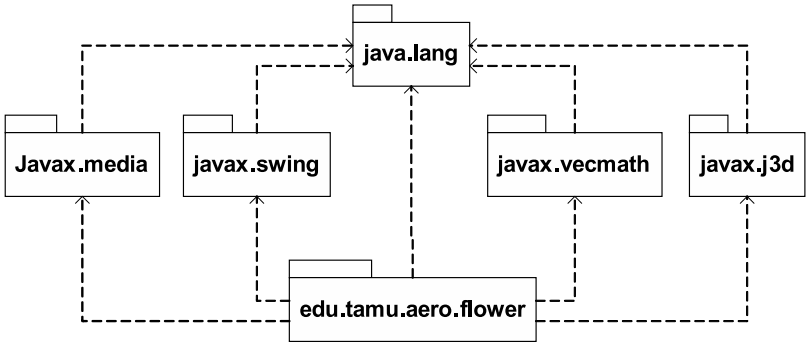


Fig. 17. Relations between the main packages. The dashed arrow means "dependency", according to the UML notation.

The `FlowerConstellation` class implements the algorithms seen in the introductory chapters. This class is responsible of creating the initial satellite distribution, computing the relative trajectory in the ECEF reference frame, and provide the orbital elements of all satellites in the constellation. Sanity checks are performed to ensure that the range of parameters remain valid. In particular, the eccentricity, semi-major axis, and perigee height are checked for consistency (e.g. prevent orbits

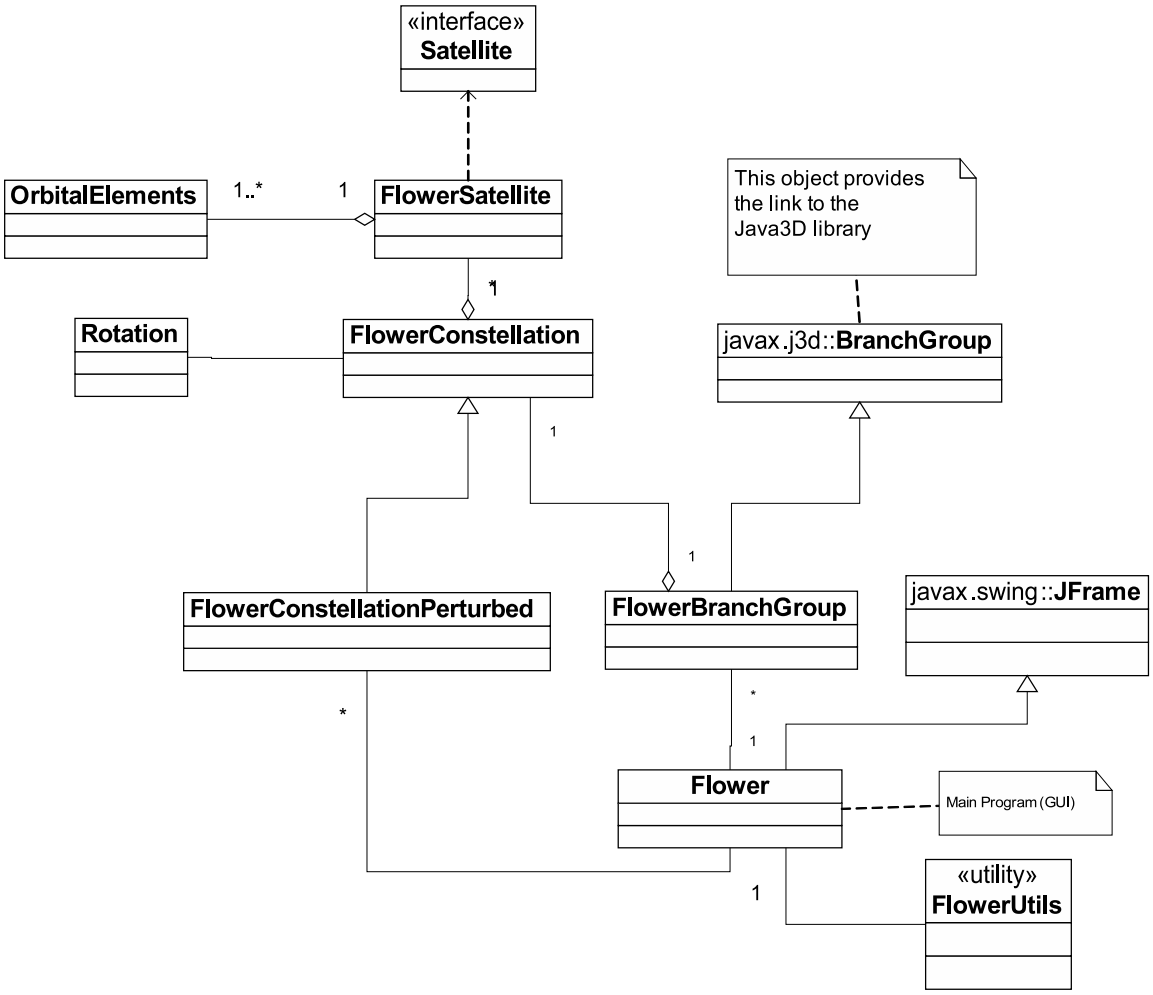


Fig. 18. Relations between the main objects, or classes, that make the FCVAT program.

"inside" the Earth).

Java3D provides a fully Object Oriented abstraction layer on top of the OpenGL API and offers higher level functionalities. In particular, it adds the `Behavior` and `Interpolator` interfaces that provide support for a great simplification of custom animation, mouse control, and interaction with the scene.

Java3D uses OpenGL to project a 3D scene onto a 2D image that is shown on a special component of the GUI called `Canvas3D`. Following the OpenGL model the

scene is built like a hierarchical tree of nodes. Each node can be a geometric primitive (e.g. point, line, polygon) or a transformation (e.g. rotation, translation, scaling, shearing, etc.). The graphic engine visits the tree from the root to the leaves, depth-first; therefore only the transformations that are encountered on the same branch of the tree of the current node have an effect on the current node itself. This fact is very important to create hierarchical models efficiently and it is amply exploited in the FCVAT to generate the 3D animation efficiently.

The *scene* tree is created by the Flower class when the user completes data entry and clicks on the *Apply* button. Once all the scene objects have been updated the Java3D thread *execute* the scene by generating the appropriate OpenGL calls, and thus updates the position of the objects on the screen. The motion of satellites on the screen however is not a simple rotation at constant angular velocity; it must follow the rules of orbital mechanics and therefore a custom interpolator, `SatelliteInterpolator` that computes the satellite position in the orbital frame has been written to implement the `Interpolator` interface for this particular type of scene object.

Once the orbital transformations, i.e. from orbital to ECI, are added to the scene tree, the only information missing to know the ECI position of a satellite at the current time is the true anomaly φ. The true anomaly, and thus the radius vector in the orbital frame, is computed by solving Kepler's equation with the Newton-Raphson method, accepting a limited precision of few decimals, since higher precision is not necessary for animation, whereas speed is. The last transformation applied is then a translation that brings the satellite in its correct position of the orbital frame. In a FC all the satellites share part of the orbital parameters (inclination, argument of perigee, eccentricity); hence some of the transformations on the scene tree can be shared, resulting in improved performance.

A simplified scene tree is shown in Fig. 19: tree nodes are depicted as ovals, light blue nodes represent transformations, light yellow nodes represent geometry (i.e. description of some scene object by lines, polygons, etc.). Euler rotations about a coordinate axis (denoted in the subscript) are represented by the symbols $R_x(\alpha), R_z(\beta)$, where the arguments $\alpha, \beta \in [0, 2\pi]$ are angles. The symbol $T(\vec{r})$ represent a translation of a vector $\vec{r}$ applied to the leaf nodes. The scene representation of Fig. 19 is necessarily simplified: many implementation details are not shown for brevity and clarity; the interested reader should refer to the Java3D documentation [57]. An example of the code needed to build the branch of the scene tree which shows the planet and the rotating orbits is given in Fig. 20.

The FCVAT has been designed so that switching between reference frames is very easy (menu `Simulation`, then select either Planet Fixed or Inertial frame). For instance, if the ECEF frame is chosen then the user will observe that the inertial orbits appear to be rotating about the Earth, while if the ECI is chosen, then it will be the relative path to appear as rotating.

It is often useful, for showing particularly interesting constellations to other researchers, interested government agencies, and for seminars. The FCVAT programs can then capture the frames and by interfacing with the program `mencoder` (part of `mplayer`, a media player under the GNU General Public License [58]) good quality video files can be generated from within the FCVAT application.

The constellation designer can add more FCs to the scene, then export the parameters to a text file, or save the constellations in an internal format for later use. Using this combination of functionalities it is possible to build constellations with hundreds of satellites and see convoluted formation flying schemes within minutes. Once the desired configurations have been found the results can be exported in MATLAB (or other programs) for further analysis.

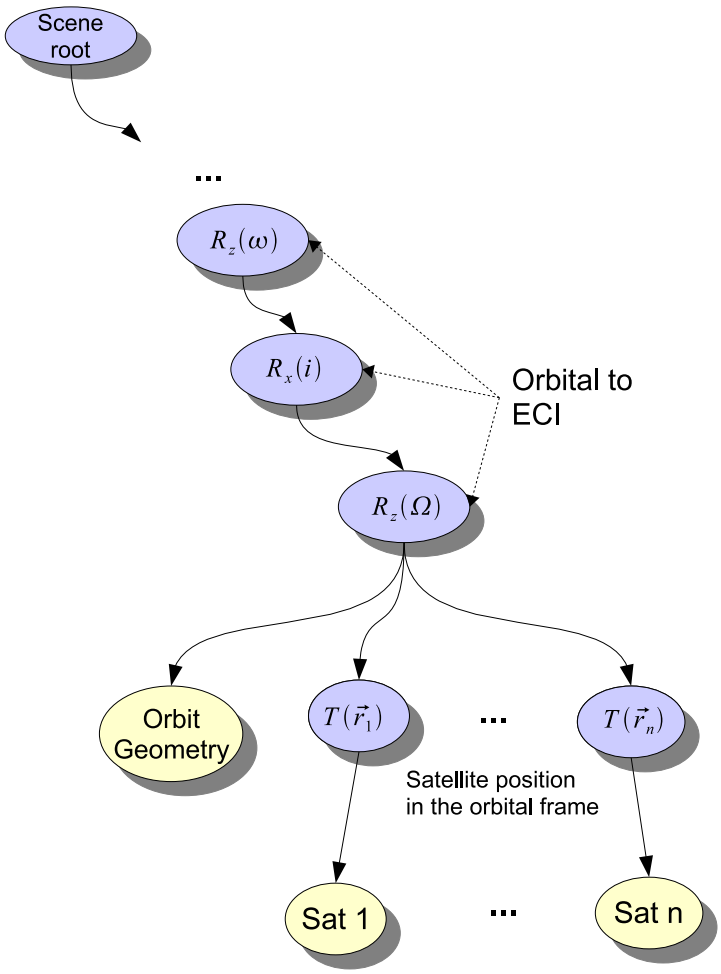


Fig. 19. Simplified representation of FCVAT scene tree for satellites belonging to the same inertial orbit (i.e. they have the same value of Ω).

```
// Subgraph for a rotating Planet on the scene
protected TransformGroup addPlanet(BranchGroup bgScene) {
    tgEarthSpin = new TransformGroup();
    tgEarthSpin.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    tgEarthSpin.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
    tgEarthSpin.setCapability(TransformGroup.ALLOW_CHILDREN_WRITE);
    tgEarthSpin.setCapability(TransformGroup.ALLOW_CHILDREN_EXTEND);
    // GAST rotation
    Transform3D rotGast = new Transform3D();
    rotGast.rotZ((float)gastAng);
    TransformGroup objSpin1 = new TransformGroup(rotGast);
    objSpin1.setCapability(TransformGroup.ALLOW_CHILDREN_EXTEND);
    objSpin1.setCapability(TransformGroup.ALLOW_CHILDREN_WRITE);
    // This rotation is added for correct alignment of the Texture
    Transform3D r0 = new Transform3D();
    r0.rotX((float) (Math.PI / 2.0));
    TransformGroup objSpin = new TransformGroup(r0);
    objSpin.setCapability(TransformGroup.ALLOW_CHILDREN_EXTEND);
    objSpin.setCapability(TransformGroup.ALLOW_CHILDREN_WRITE);
    objSpin1.addChild(objSpin);
    tgEarthSpin.addChild(objSpin1);
    // ECEF axes
    objSpin.addChild(new Axis((float) planet.getEquatorialRadius() * 1.5f));
    if (bgPlanet != null)
        bgPlanet.detach();
    bgPlanet = planet.makeBranchGroup();
    bgPlanet.setCapability(BranchGroup.ALLOW_DETACH);
    objSpin.addChild(bgPlanet);
    alphaEarth =
        new Alpha(-1,
            (long) (Math.abs(planet.getRotationPeriod())
                * 1000.0 * warpFactor));
    if (!animation) alphaEarth.pause();
    earthInterp = new MyRotationInterpolator(alphaEarth, tgEarthSpin);
    earthInterp.setEnable(animation);
    // make it rotate about z axis
    Transform3D zSpin = new Transform3D();
    zSpin.rotX((float) (Math.PI / 2.0));
    earthInterp.setTransformAxis(zSpin);
    earthInterp.setSchedulingBounds(
        new BoundingSphere(ORIGIN, (float) planet.getEquatorialRadius()));
    earthInterp.setEnable(refFrameType == PLANET_CENTERED_INERTIAL);
    bgScene.addChild(earthInterp);
    return tgEarthSpin;
}
```

Fig. 20. Example of code used to generate the scene graph.

B. The Flower Constellation Matlab Toolbox

MATLAB is an ideal platform for rapid prototyping and for the quick visual analysis of results. Its powerful numerical language (based on the BLAS and LAPACK FORTRAN libraries) coupled with its superb plotting functions makes it an ideal workbench for modern engineers.

While working on understanding FCs and building the software for their analysis, a large number of small programs had been incoherently written: many code snippets were copied from one program to the next generating redundancy and inconsistency, arguments were passed with global variables and comments were scarce, quickly generating a debugging nightmare. The need for a general reorganization of the MATLAB code pertaining FCs and orbital mechanics available was recognized and the challenge met. The result of this general code overhaul is the FCToolbox for MATLAB.

The FCToolbox contains:

- Functions for generating Flower Constellations in a number of different ways.
- The interfacing code to read and write output from (or for) the FCVAT program.
- A number of utilities for plotting FCs with MATLAB in 3D and on 2D maps.
- Functions for transformation of coordinates between different reference frames and from geocentric to geodetic coordinates.
- Orbital mechanics and propagation routines.
- Functions for Dilution of Precision (DOP) analysis.
- Functions for Coverage and revisiting time analysis.

- Time related routines, i.e. Julian Date, GAST angle computation.
- Utilities for the generation of high quality 3D plots of Earth orbiting satellites.
- Test programs for the routines in the toolbox itself

The logical structure of the toolbox is shown in Fig. 21.

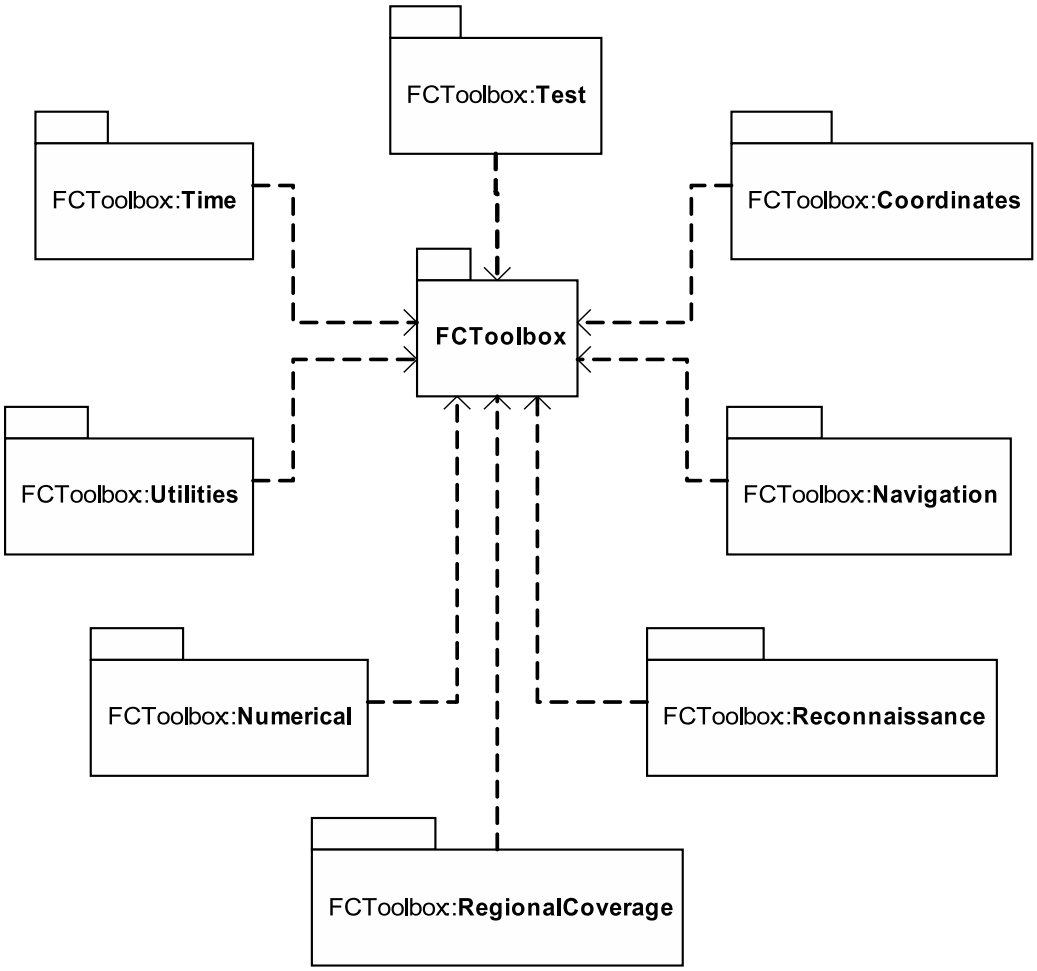


Fig. 21. Logical structure of the FCToolbox.

MATLAB support for OOP is still in its infancy: the updates to the language are introduced very slowly because many MATLAB users are only casual programmers and are not interested in building big, modular programs. In any case the

little that is available is useful and the FCToolbox contains three classes that are used to greatly simplify the creation of FCs with MATLAB: `FlowerConstellation`, `OrbitalElements`, and `Planet`, see appendix A for more details.

The FCVAT application and the FCToolbox have been used to develop and test the rest of the code. They proved to be important tools to design Flower Constellations for some specific mission concepts shown in the following chapter.

CHAPTER VI

APPLICATIONS

The problems examined in this chapter have been chosen to represent diverse and significative samples of the range of missions for which constellations of satellites have been used in the past or it is foreseeable that they might be used in the future.

During the time of the writing of this dissertation the Congress of the U.S.A. approved the release of high resolution images acquired from space to civilian federal organizations, like the FBI, for Homeland Security purposes. So far only military agencies and the CIA have been officially authorized to access some of the higher resolution images. The purpose of this change of policy is that of monitoring ports and sensible facilities to prevent terrorist attacks and to provide critical intelligence for the coordination of rescue efforts in case of natural disasters, like the flooding of New Orleans, LA, in August 2005, which was caused by the rupture of the levies during the landfall of hurricane Katrina on the Louisiana coast.

Another application of great interest is global navigation: how well FCs could be used to reproduce or provide new solutions for the space segment of existing (GPS) and planned (GALILEO) global navigation systems is also examined.

Before entering in the mission details some more definitions regarding Earth observation and some important system engineering issues regarding mission design must be introduced in order to motivate the choice of the particular orbits shown in the example missions.

A. Geometry of Earth Observation

The reference ellipsoid has an equatorial radius of $R_\oplus = 6,378.137$ Km and a flattening factor, $f \simeq 0.0033528$. A more precise approximation can be obtained by

measuring the Earth gravitational field and representing the Earth as an equipotential mean surface as if the oceans were extended over the continents: this procedure gives the geoid. Space-based measurements of the gravitational anomalies of the Earth and the Moon have been done by both U.S. and former U.S.S.R. in the early stages of the space age in order to model the gravitational perturbations and thus improving spacecrafts navigation capabilities[1].

For practical purposes concerning this dissertation, the approximation of the Earth surface with a reference ellipsoid (WGS84)[2] is utilized in the following chapters, unless otherwise noted, since the difference between the reference ellipsoid and the geoid is always less than 200 m [59]. A graphic depiction of the relation between geocentric and geodetic coordinates is shown in Fig. 22. Eq. (6.1) allow the transformation from geodetic coordinates,, i.e. geodetic latitude ϕ_g, longitude λ_g and altitude h, to cartesian geocentric coordinates (i.e. ECEF, x, y, z). The inverse transformation requires an iterative approach or approximations techniques as in [60, 61].

$$x = (N(\phi_g) + h)\cos\lambda_g \cos\phi_g \tag{6.1a}$$

$$y = (N(\phi_g) + h)\cos\lambda_g \sin\phi_g \tag{6.1b}$$

$$z = [N(\phi_g)(1 - e^2) + h]\sin\lambda_g \tag{6.1c}$$

$$N(\phi_g) = \frac{a}{\sqrt{1 - e^2 \sin^2\phi_g}} \tag{6.1d}$$

In eq.'s (6.1) the term $N(\phi_g)$ is the curvature of the reference ellipsoid at the given latitude. The terms a, and e are the semi-major axis and the eccentricity of the reference ellipsoid respectively.

Some definitions that can be found also in [62] pertaining to Earth observation

[1]Not to mention the accuracy of ICBMs, since the motivation of most of this research was, unfortunately, the *cold war*.

[2]Department of Defense, World Geodetic System, 1984 standard

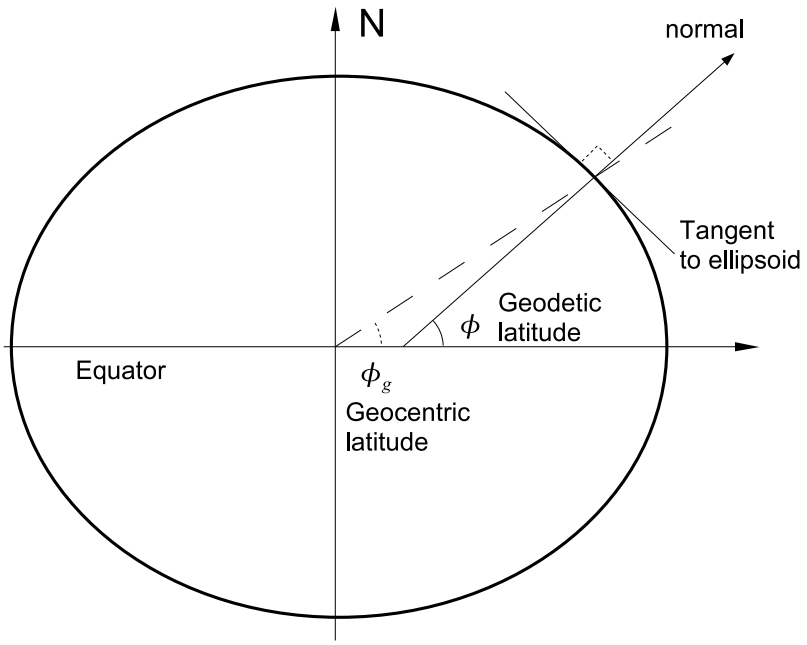


Fig. 22. Geodetic and geocentric latitude; note that oblateness of the Earth ellipsoid has been exaggerated for clarity.

will turn to be useful in the following.

Ground Track The *ground track* is the projection of the satellite trajectory on the Earth ellipsoid.

SSP The *Sub Satellite Point* (SSP) is the point on the Earth ellipsoid that sees the satellite exactly perpendicular above, i.e. with an elevation of 90°.

Nadir The vector originating from the satellite position, perpendicular to the Earth ellipsoid (i.e. pointing perpendicularly to the SSP) is called *Nadir*.

Zenith The opposite vector, from the SSP to the satellite, perpendicular to the Earth ellipsoid, is called *zenith*.

Access Area The portion of the Earth surface seen by the satellite at a given time with some sensing instrument is called the *Access Area*.

The geometry and the symbols are better described by Fig. 23. It is easy to see that the size of the access area depends from the altitude h and from the sensor field of view, defined for a conical sensor by the off-nadir angle η. The angle corresponding to the footprint of the sensor around the SSP is called *Earth central angle*, and it is denoted with λ. The Earth central angle reaches a maximum, λ_{max} value when the sensor cone becomes tangent to the Earth surface. The angle from the local horizon to the satellite is called *elevation angle*[3] and is denoted with ε. The *swath* is defined to be the envelope of the satellite footprint as the satellite travels in its orbit. The resulting portion of Earth surface observed looks like a stripe (for LEO satellites), called indeed *swath*.

The Matlab code used to compute the Earth central angle[4] is given in Fig. 24, note that this code assumes a spherical Earth of equatorial radius R_e, and supports vector input for maximum efficiency, as recommended by Mathworks.

B. Mission Design for Satellite Constellations

Since the satellite footprint size varies with altitude this is another issue that is often mentioned to criticize elliptical orbits: designing a sensor for remote sensing becomes more complicated because of the varying resolution that, at least for camera type sensors, is obviously a function of distance; thus should the sensor be designed to work at perigee, at apogee, or both? A competing set of requirements often translates in more weight, more power, and more space: the three fundamental cost drivers for payload launch costs.

The use of elliptical orbits in a constellation can also be seen as an opportunity:

[3]Often called *grazing* angle.

[4]In the code the angle λ_{max} has been renamed ρ

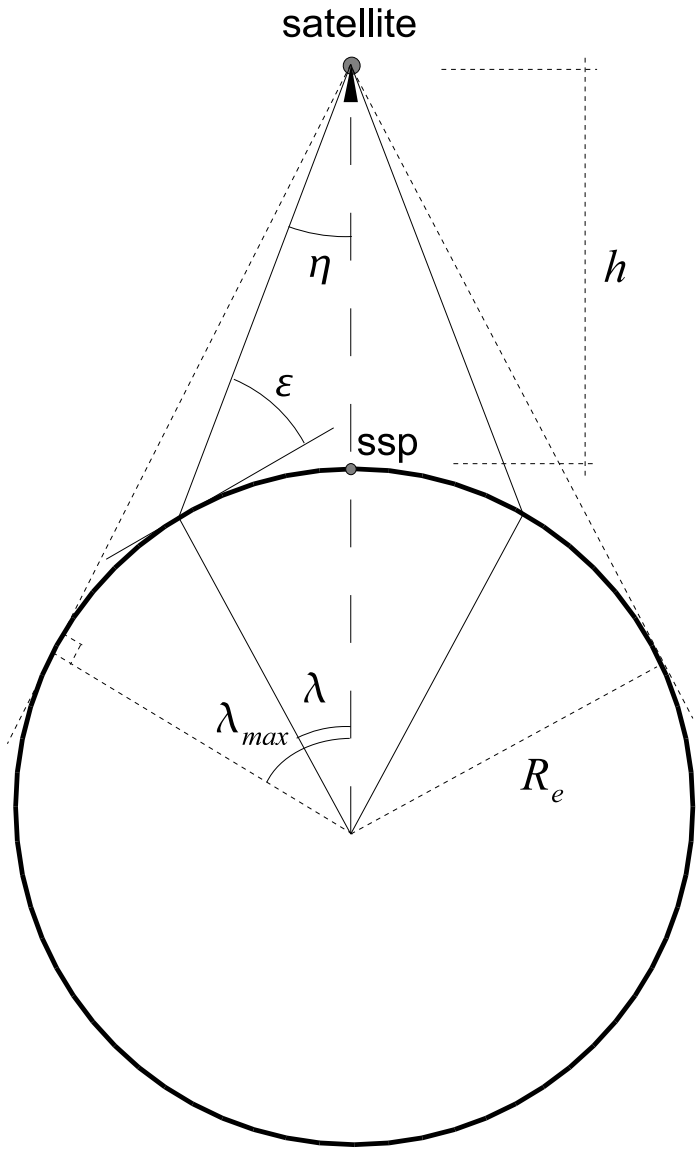


Fig. 23. Geometry of Earth observation from satellite.

```
function [lambda, swath] = swath_ang(range, eta)
% Compute the Earth Central Angle and swat given the range
% and the sensor half FoV angle.
% Synopsis:
%   lambda = swath_ang(range, eta)
%
% Input:
%   range  Altitude of the satellite in Km from Earth center.
%   eta    Sensor half field of regard (angle from nadir), [rad].
%
% Output:
%   lambda Earth central angle, i.e. angle on the Earth surface
%          corresponding to the half field of regard of the sensor
%          from the SSP,
%          measured from the center of the Earth in [rad].
%   swath  Swath in Km on the reference sphere of radius Re
%
% Note:
%   This function accepts vector input.
%
% See also SWATH_PATCH, SWATH_LINES.
% $Author: Christian Bruccoleri$ $Date: 07/28/07$

    % maximum visible angle at given
    %altitude range
    sin_rho = Re ./ range;
    sin_eta = sin(eta); % sin of the half sensor angle
    cos_elev = sin_eta ./ sin_rho; % cos of elevation angle

    lv = abs(cos_elev) <= 1; % find valid values of cos(elev)
    % nominal case: a fraction of the earth sphere is seen
    elev(lv) = acos(cos_elev(lv));
    lambda(lv) = pi/2 - eta - elev(lv);

    % cases for which eta is too big, i.e. all emisphere
    % is visible lambda = lambda_max
    % cos(lambda_max) = sin(rho);
    lambda(~lv) = acos(sin_rho(~lv));

    % compute full swath width in km
    swath = 2*Re*lambda;
end
```

Fig. 24. MATLAB code used to compute the Earth Central Angle.

dual use is an expression that is becoming ever more popular thanks to the miniaturization of electronics components. It is now possible to think about sensors that can perform several tasks without paying a very high price in power consumption and payload weight.

Van Allen Radiation Belts

The most serious long term threat to the survivability of the spacecraft electronics is radiation. The Earth is surrounded by two toroidal regions of charged particles: the first from $1,000$ km to $10,000$ km of altitude and the second from $13,000$ to $65,000$ km. These regions are called the Van Allen belts[5] and were discovered at the beginning of the space age, in 1958, during the Explorer I mission. Their existence however had been suggested before by theoretical models.

The regions are not uniformly charged: the level and type of radiation changes with the altitude and the polar regions are practically swept clean of particles as it can be seen in Fig. 25 [63] and Fig. 26 [64]. Their formation is due to the interaction of the Earth magnetic field and the high energy particles coming from the Sun and from cosmic rays.

The inner belt is constituted mainly by high energy protons (> 50 MeV and even exceeding 100 MeV), believed to be caused by β-decay of neutrons resulting from collisions between cosmic rays and the upper layers of the Earth atmosphere. The closest approach of the inner belt to Earth is the South Atlantic anomaly, at about 200 km of altitude.

The outer belt is mainly constituted by electrons trapped by Earth's magnetosphere, but contains ions too: alpha particles and high energy protons. Non shielded

[5]From the name of the scientist who studied them first, Dr. James Van Allen, University of Iowa.

electronic instruments must be turned off during the time in which a spacecraft crosses the Van Allen belts in order to reduce the risk of permanent damage.

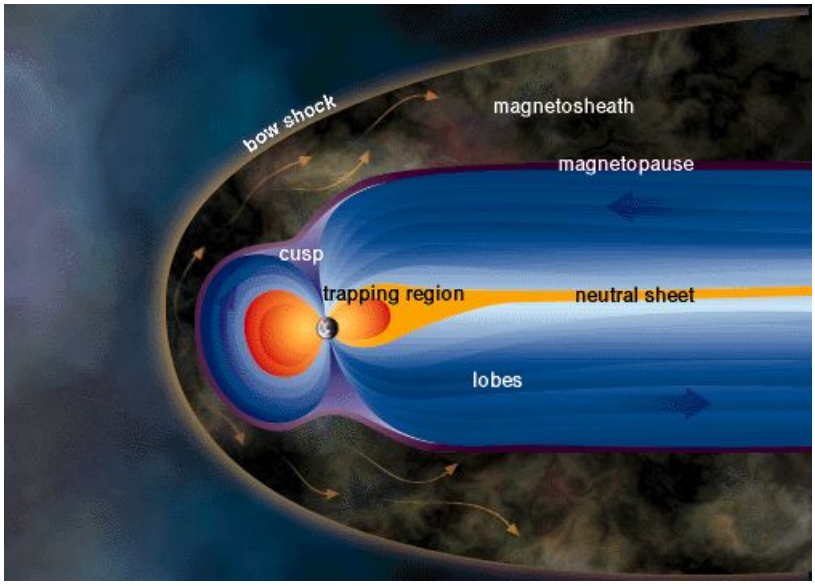


Fig. 25. Van Allen Belts: formation process [63].

The Van Allen belts have an important impact in mission design: in fact the three broad classes of Earth orbits used for most missions are defined with respect to them. Low Earth Orbits (LEO), if the altitude is less than $1,000$ km, thus below the first belt. Medium Earth Orbits (MEO) for altitude between $10,000$ and $13,000$ km, i.e. between first and second belt, and GEO for altitude around $36,000$ Km, for which the radiation becomes tolerable again with appropriate shielding.

Since the requirements on radiation shielding play such an important role in the selection of components for space flight and thus, ultimately, on cost and lifetime, for most missions one is restricted to operate outside the highly charged regions of the Van Allen belts. This consideration has been taken into account for the examples shown in the next sections.

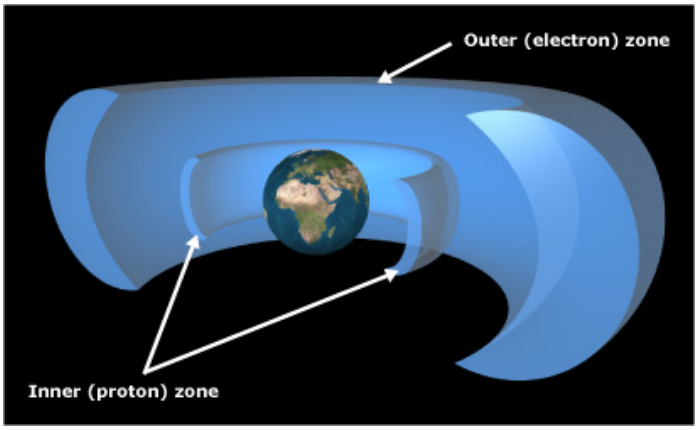


Fig. 26. Approximate shape of the Van Allen's Belts: inner belt (mainly protons) and outer belt (mainly electrons) [64].

Number of Launches and Constellation Deployment

Minimizing the number of launches to deploy an assigned constellation is achieved by simultaneously launching as many satellites as possible with the same launcher. The minimum number of launches is bounded by the carrier payload capacity, since the number of satellites is assigned during the constellation design.

Once the parking orbit has been reached, the deployment of the constellation satellites into the final orbit should be completed in the minimum amount of time. The difference in nodal precession rates at different altitudes is usually adopted to align different orbital planes. Depending on whether the constellation is LEO type (mostly circular orbits) or MEO/HEO (circular or elliptical orbits), two distinct approaches are adopted:

LEO Consider a constellation with 4 satellites in 55° inclination, and 600 km altitude with circular orbits. The orbital planes are evenly separated by $\Delta\Omega = 90°$. For this LEO constellation a Space Shuttle carrier deploys one satellite, then lowers its orbit by h, lets the right ascension change until the orbit plane is aligned with the next desired plane (90° relative change) then maneuvers to back to the

constellation orbit, drops the next satellite, and then continues the process.

MEO/HEO Consider deploying a 4-satellite Broglio-type FC with two launches, critically inclined, whose satellites must be in 4 orbits that are displaced by $\Delta\Omega = 90°$. Consider a launch taking 2 satellites in a parking orbit (e.g.: 500 km circular and critically inclined). One S/C maneuvers into the nominal FC orbit while the other remains in the parking orbit. Because of the J_2 perturbation, the two orbits precede with different RAAN rates.

For a parking orbit with altitude $h_p = 500$ km, we have $\dot{\Omega}_p \approx -3.42578$ deg/day, while for a Broglio-type nominal orbit ($a = 20,270.42$ km, and $e = 0.66068$), we have $\dot{\Omega}_{FC} \approx -0.245527$ deg/day. Therefore, only 28.3 days are required to reach a 90° RAAN difference.

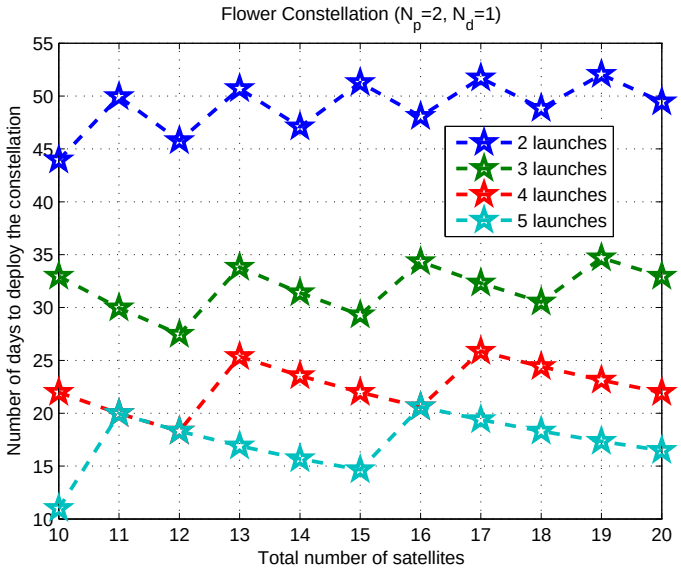


Fig. 27. Launches vs deployment time for a 2/1 FC.

Figures 27 and 28 summarize the number of days to deploy a 2/1 and a 3/1 FC, respectively, as a function of the number of launches (carrier capacity) and with the

constraint of having the satellites uniformly distributed on the full 360° RAAN range.

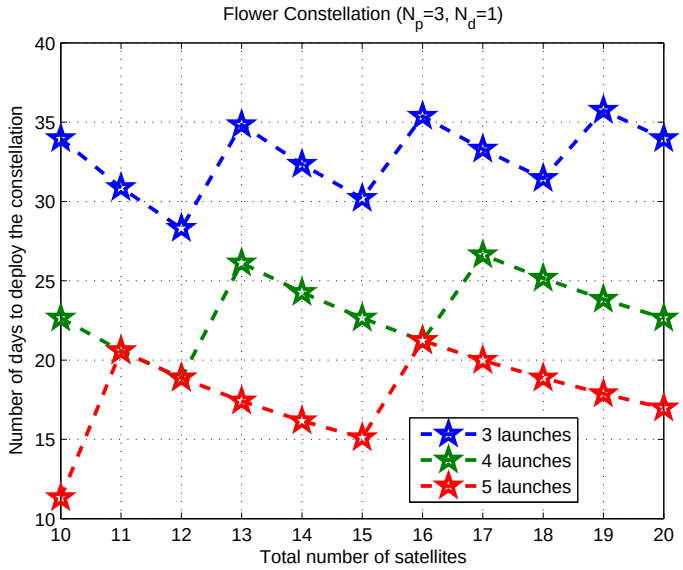


Fig. 28. Launches vs deployment time for a 3/1 FC.

C. Global Coverage

Global Coverage is the problem of observing from space the whole Earth surface within a specified interval of time. Moreover one may be interested in a minimum *revisiting time*, i.e. the time between two passages above the same point or region. This requirement is usually met by using polar orbits, thus by using the Street of Coverage design method. Flower Constellations are particularly well suited for this kind of requirements since the use of compatible orbits makes it possible to design a constellation that provides continuous coverage with a specified revisiting time.

Space Uniform Satellite Distributions

The first problem presented concerns a FC in which the satellites are spread uniformly on the relative path, such that they always remain at a fixed distance from each other. Such scheme translates into non-intersecting access areas, and has the additional benefit of minimizing any risk of collision between satellites.

Even though space is big and a satellite is comparatively very small, the risk of collision in a constellation, especially over a long period of time, should not be underestimated[8]. If a spacecraft malfunctions it could start to drift because of orbital perturbations, eventually coming to interest more closely the orbits of other spacecrafts. If there is an explosion, either because of a malfunction or because of a deliberate attack, in which a great number of small debris are generated, it is possible that these, with time, spread over the whole orbit[6] and thus the chances of collisions are greatly increased.

Therefore an optimal satellite constellation for continuous global coverage can be designed by having the SSPs uniformly distributed over the Earth surface. Therefore the problem is similar to that of uniformly distributing points on a spherical surface, which is also known as the *Thompson problem*.

Closed form solutions to this problem exist only for few points. Using Genetic Algorithms, it is possible to find an approximate solution to the problem of optimally distributing the spacecraft along the relative path by minimizing the overall potential energy associated with particle charges (satellites) at any instant of the repetition time. Genetic Algorithms have been adopted because six of the FC design parameters are integers and this is a situation where classic analytical optimization techniques cannot be used.

[6]Much like Saturn's rings.

Fitness functions

Satellites have been treated like charged particles orbiting the Earth and the potential energy associated to their spatial configuration at each instant in time has been computed. The potential energy function utilized is of the form shown in eq. (6.2):

$$U(t) = \sum_{i=1}^{N_s-1} \sum_{j=i+1}^{N_s} \frac{\chi}{\|\vec{r}_{ij}(t)\|} \tag{6.2}$$

where $\vec{r}_{ij}$ is the vector from satellite j to satellite i, N_s is the number of satellites in the constellation, and χ is a constant that can be chosen arbitrarily for this case and that should be scaled to give the desired numerical stability.

The maximum value of the potential energy $U(t)$ over the constellation repetition time is chosen as the cost, denoted with J, of the configuration under scrutiny since it is the value associated with the instant in time in which satellites are closest:

$$J = \max_t U(t) \tag{6.3}$$

In the implementation, a different fitness function has been utilized: instead of using the distance between the satellites i and j, $\|\vec{r}_{ij}(t)\|$, the angle between the radii, α_{ij} has been used because it was much more efficient to compute with Matlab and achieves the same objective. If one builds a matrix $R = [\hat{r}_1(t) \dots \hat{r}_N(t)]$ of the unit vectors corresponding to the directions of the radii of all N satellites, then the matrix of the cosine of the angles between the satellites can be simply computed as $R^T R$.

Although this approach introduces redundance (the angles of interest are only the ones on the upper or lower diagonal, indifferently) this method of computation is much faster with Matlab because makes much more use of built-in functions. By making the computation of the cost function faster one allows the GA to span the space of admissible solution much more in depth.

Summarizing, the fitness function chosen for this problem involves the following steps:

1. Decode the chromosome and extract the FC parameters corresponding to the individual in input.
2. Propagate the position of each satellite for the duration of the FC ECEF period.
3. Compute the function $U(t)$ from eq. (6.2) at each instant in time.
4. Choose as the cost of this configuration the value given by eq. (6.3)

Of these steps, the chromosome encoding/decoding is the part on which more attention must be put because of efficiency issues. As explained in chapter IV, GAs can not directly satisfy constraints on the input variables, therefore a straightforward encoding of the FC parameters as genes of the chromosome would lead to the generation of a very large number of unfeasible or redundant solutions since there are several constraints on the admissible integer parameters of a FC, as seen in chapter III.

Alternatively, one should rewrite the genetic operators in such a way that unfeasible solutions are not generated. This road has been tried, but the improvement obtained was in the end very marginal, since the implementation of the operators had to take the burden of enforcing the constraints at every step and therefore a large number of computations were still needed.

A better way to circumvent the problem was found in a different encoding of the chromosome. The number of satellites in the constellation N_s and the maximum repetition time, let it be $N_d, \max$ are fundamental design parameters: therefore they are not object of optimization, they are provided as user input.

Given this user input it is then possible to pre-compute all the unique FCs admissible configurations of the 6 integer parameters and save them as the columns of a matrix of integers, let it be called F_c[7]. One can have a gene in the chromosome that represents the column index of the configuration in F_c, i.e. simply an integer number. The constraints are enforced by the algorithm that generates the admissible solutions, only once as an initialization step and all the six integer parameters are encoded with only one index. Furthermore, once the matrix F_c for the given $N_d, \max$ and N_s has been computed it can be saved and re-used to solve different problems, thus sensibly reducing the overhead.

GA parameters

The optimization has been carried on with the GA parameters shown in Table. II. The meaning of these parameters is briefly explained below and should be clear from the introduction given in chapter IV.

Population Number of individuals in the population.

Mutation rate Probability of mutation.

Crossover rate What portion of the population should be renewed at each generation by using crossover.

Generations Maximum number of iterations allowed before stopping.

Results

A number of configurations has been generated by varying the number of satellites from 4 to 10. Tables III and IV show the parameters of these configurations for

[7]With uncommon display of imagination.

Table II. Values of the main GA parameters used for global coverage.

Parameter	*Value*
Population	40
Mutation rate	0.6
Crossover rate	0.2
Generations	30

Table III. Prograde FC configurations for the global coverage problem. A value of '*' in the ω column means that the orbit is circular.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
18	5	4	1	4	3	11,572	41.2	*
11	3	5	2	5	1	11,354	48.9	*
37	10	6	1	6	7	11,247	58.2	*
17	5	7	3	7	2	12,270	57.9	*
18	5	8	3	8	2	11,572	57.0	*
19	5	9	7	9	3	10,937	61.3	*
24	7	10	3	10	4	12,166	62.3	*

prograde and retrograde inclinations respectively. An orbit is said to be prograde if the inclination is less than 90° retrograde if inclination is greater than 90°[8].

The ground track and the access area of some of the configurations in III and IV are shown in Figs. 29, 30, 31, and 32; For each configuration a video file has been generated using the FCToolbox and it is available for download at the Flower Constellation web site: http://flowerconstellations.tamu.edu

The choice of the sensor field of view which, together with the altitude of the

[8]If the inclination is exactly 90° the orbit is polar.

Table IV. Retrograde FC configurations for the global coverage problem.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
26	7	4	3	4	5	11,202	130.7	*
29	8	5	2	5	6	11,490	130.4	*
4	1	6	1	6	0	10,355	125.0	*
31	9	7	3	7	6	12,109	121.8	*
10	3	8	5	8	2	12,517	122.0	*
17	5	9	1	9	2	12,270	117.4	*
32	9	10	3	10	3	11,722	117.3	*

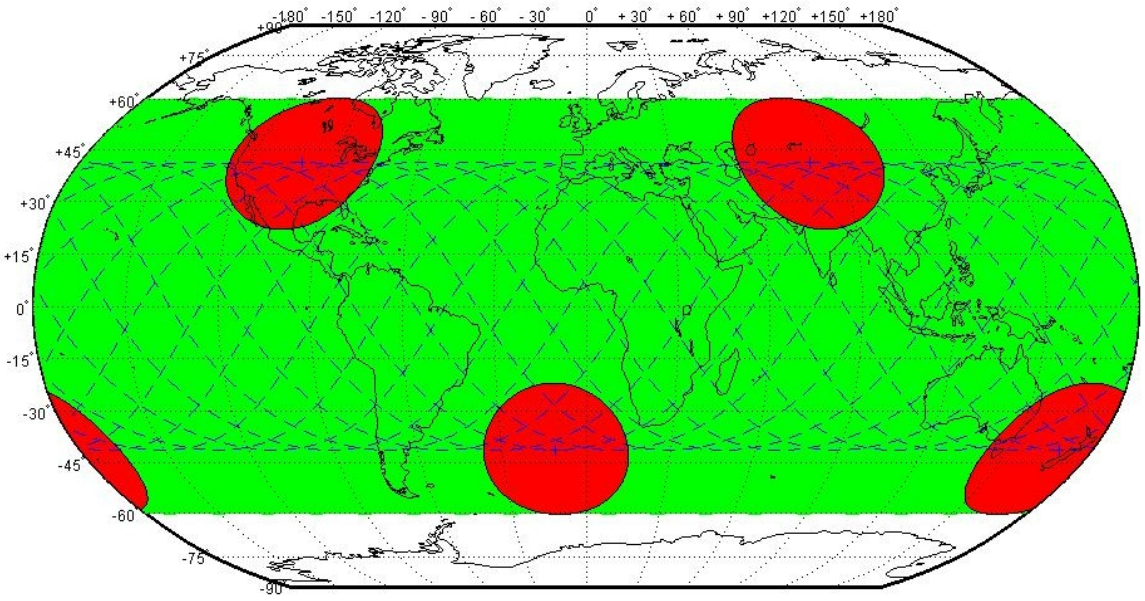


Fig. 29. Uniform prograde FC with 4 satellites; snapshot at time $t = 0$.

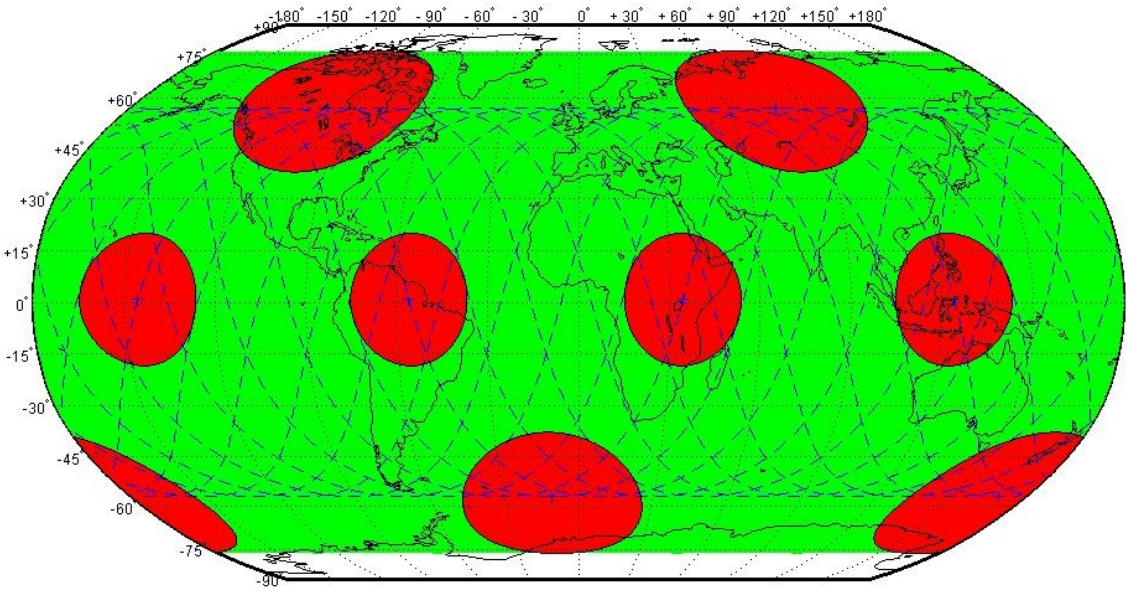


Fig. 30. Uniform prograde FC with 8 satellites; snapshot at time $t = 0$.

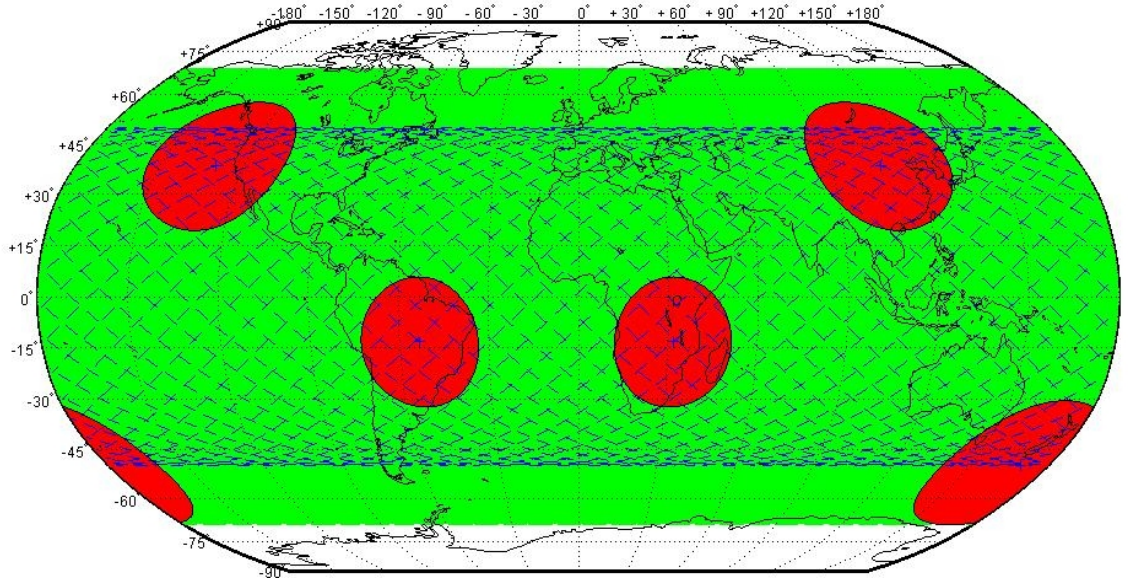


Fig. 31. Uniform retrograde FC with 5 satellites; snapshot at time $t = 0$.

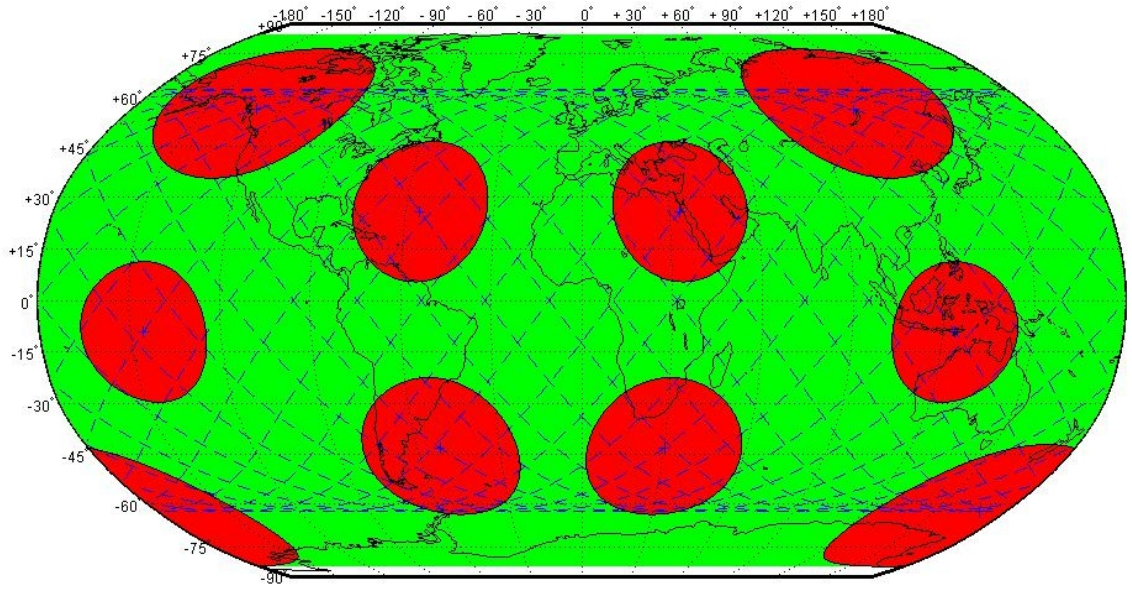


Fig. 32. Uniform retrograde FC with 9 satellites; snapshot at time $t = 0$.

constellation, determine the size of the access area, depends on the sensor, which in turn depends on what the mission profile is. Global coverage may be useful for communications, for weather monitoring, or for various applications of hyperspectral imaging, which will be discussed in more details in the section on the reconnaissance problem. The value chosen for the examples in this section is a conic sensor with the off-nadir angle $\eta = 10°$. It can be seen that these examples achieve global coverage of all the inhabited regions; the polar region are not covered.

Summarizing, the Uniform FC configurations presented in this section meet the global coverage requirement. The choice of the revisiting time, ΔT_r, depends on the number of spacecrafts in the constellation: $\Delta T_r = T/N_s$, with T being the periodicity of the constellation. This property holds for constellations in which the satellites are uniformly distributed in time along the relative trajectory, as the configurations presented in this section all are.

Time Uniform Satellite Distribution

The concept of uniformly (with respect to time) distributed satellites on the relative path can be used also to provide coverage of one hemisphere, i.e. northern or southern, with critically-inclined highly-elliptical orbits (HEO). This concept has been one of the first envisioned applications of Flower Constellations, and it is included here for completeness and to provide a further example of the capabilities of both the FCToolbox and of the FC design methodology.

The very first FCs to be analyzed was the Broglio FC, named after Gen. Luigi Broglio[9] who first designed a very similar orbit concept, with the main difference being the equatorial inclination; Gen. Broglio's Sistema Quadrifoglio (eng. "Clover System") was never implemented, and no attempt to generalize the idea to include other possible constellations were made at the time. I never had the honor to meet Gen. Broglio, but he had been my advisor's advisor and so a small part of his legacy passed onto this work.

A FC simulating the original parameters of "Sistema Quadrifoglio" is shown in Table V, and its appearance with the ECEF trajectory visible is shown in Fig. 33 as simulated by the FCVAT program. With small modifications, this can become

Table V. Original parameters of "Sistema Quadrifoglio" simulated with FCs.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
4	1	4	1	4	0	600	0	270

a system for continuous coverage of the whole northern hemisphere with 4 satellites, as shown in Table VI. The coverage characteristics are shown in Fig. 34: three out of four satellites are always in view of the northern hemisphere and are spread 120°

[9]Founder of the "Centro Ricerche Aerospaziali (CRA), San Marco," Rome, Italy; NASA partner and which launched several operational satellites from Kenya.

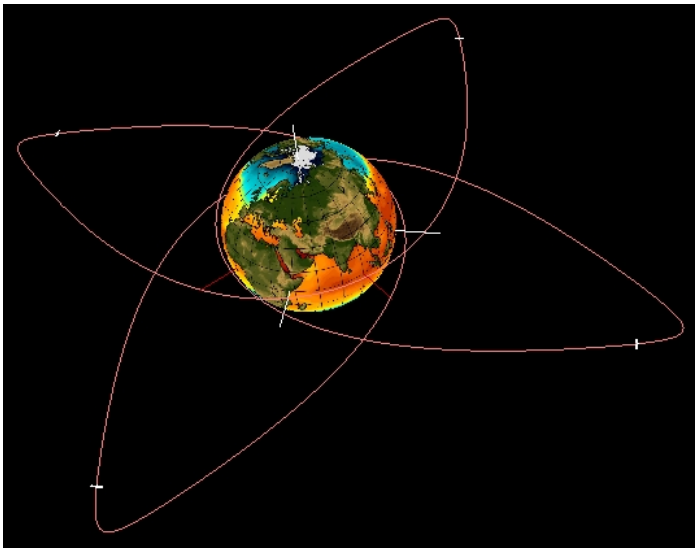


Fig. 33. "Sistema Quadrifoglio"-like FC simulated with FCVAT.

Table VI. FC for continuous persistent coverage of the northern hemisphere with 4 satellites.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
3	1	4	1	4	0	600	63.4	270

apart. There is a juggling effect between a satellite leaving the apogee and another taking its place that guarantees the persistent coverage. The northern hemisphere has been chosen simply because it is more densely inhabited, but changing the FC to observe the southern hemisphere is only a matter of changing the argument of perigee, ω, to 90°. The access area of one satellite is partially overlapping with that of its neighbors, but this fact depends on the choice of the sensor field of view. The off-nadir angle defining the conical sensor has been chosen for this example to be $\eta = 10°$.

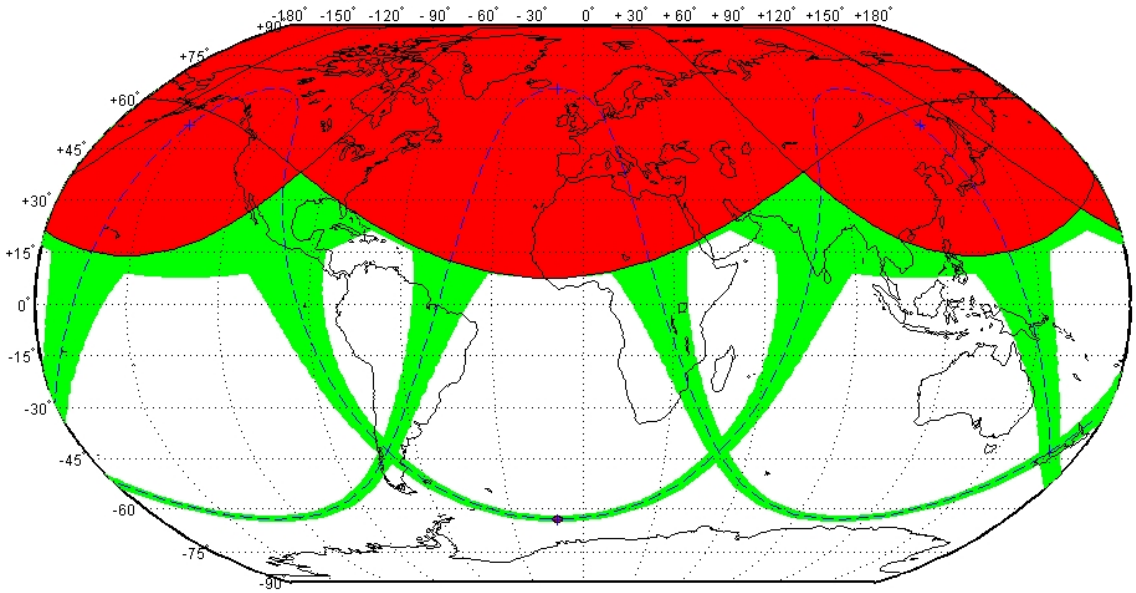


Fig. 34. Coverage of the northern hemisphere by a Broglio FC: 3 satellites are close to apogee whereas one satellite is at perigee. The relative motion is such that three satellites are always in view of the northern hemisphere.

Flower Constellations for Global Navigation

If there is a satellite constellation of which most of the people, even those not acquainted with technical matters at all, are probably aware of, that is the GPS constellation. Global Navigation, thanks to the vast number of small, inexpensive, and reliable GPS navigation devices, is probably second only to satellite television with respect to number of users of a space based system. For this reason the possibility of applying FCs to global navigation must be studied.

A previous work, by Park et Al. [65] laid the basis for the use of FCs for Navigation. The constellation presented in that work however had been found by inspection and intuition. In this dissertation a more systematical approach to finding an optimal constellation for navigation has been used. In both works, the underlying idea is that in order to provide a good position estimation a FC must be symmetrical

with respect to the Earth axis.

The fundamental parameter that measures the quality of position estimation for navigation systems is the Geometric Dilution of Precision (GDOP). The position of a receiver on the Earth surface by GPS signal is similar to the problem of triangulation. The receiver gets a signal from each GPS satellite in view from which the pseudo-range can be computed.

The signal contains the ID of the satellite, the time of the signal emission, and the orbital elements (i.e. position) of the emitting satellite itself. At this point it is possible for the receiver to compute the pseudo-range by using a Gaussian Non-Linear Least Square method, that is nothing more than the 3D version of the triangulation method used for finding the position of a ship in the ocean by using signals emitted by known ground stations. This algorithm, used since the dawn of radio transmissions, uses the position of at least three known emitting stations and computes the distance to each station as $c\,\delta t$, where c is the speed of light; the intersection of (at least) three circles provides the receiver position.

Since the GPS receiver must estimate 4 quantities, position x, y, z of the receiver and time of flight, δt, at least four measurements are needed. Naturally this process is affected by errors; thus, in general, the more measurements (i.e. satellites in view) the better the position estimation will be.

The GDOP is defined as the covariance of the sensitivity matrix H:

$$H = \begin{bmatrix} \hat{s}_1^T & 1 \\ \vdots & \vdots \\ \hat{s}_N^T & 1 \end{bmatrix} \tag{6.4}$$

where $\hat{s}_i$, for $i = 1..N$ are the line of sight (LOS) vectors from the receiver to the i-th

satellite, expressed in the topocentric reference frame, i.e. NED frame[10], and N is the number of satellites visible from the receiver. Thus,

$$GDOP = \sqrt{tr\ (H^T H)^{-1}} \tag{6.5}$$

and the covariance of the position error is

$$\sigma_p = \sigma_R\, GDOP \tag{6.6}$$

where σ_R is the covariance of the pseudo-range error estimation, where the assumption that the errors in pseudo-range $R = c\delta t$ are uncorrelated and are caused by zero-mean gaussian noise.

Since the GDOP is computed as in eq. 6.5, it comes natural to ask, what if the matrix $(H^T H)^{-1}$ can not be inverted? In this case the GDOP is undefined, and in practice it means that the configuration of the visible satellites is bad for position estimation. The limit case is if all the satellites are seen close to the zenith: the LOS vectors will appear to be almost the same and thus the result is a poor GDOP (the lower the better, the best possible value is 1). In practice a $GDOP > 6$ means that the position prediction is unreliable. That happens in practice too and that is why it is usually better to have more than the minimum 4 satellites in view when estimating position with GPS.

This case can be easily understood by going back to the triangulation problem: if the emitting stations are aligned with the receiver there are multiple possible solutions and therefore the position of the receiver can not be determined.

Summarizing, what is important for position estimation using GPS is:

- The number of GPS satellites visible from a ground receiver at a given time;

[10]A reference frame centered on the receiver, tangent to the Earth with the axes pointing in the North, East and Down (Nadir) directions.

Table VII. Parameters of existing constellations for global navigation.

Constellation	*# Sat.*	*Altitude* [km]	*Incl.* [deg]	*Walker t/p/f*
GPS	24	20,184	55	-
GalileoSat	27	23,616	56	27/3/1
GLONASS	24	19,100	64.8	24/3/1

- Their geometrical configuration (or GDOP).

The Galileo and GLONASS constellations are very similar in concept to the GPS and provide similar accuracy in position estimation. Galileo is the newest and when deployed should provide the best accuracy, but studies are already being made for using both systems at the same time, thus achieving an even better accuracy than by each set of satellites alone.

All the navigation constellations are based on equally spaced circular orbits at similar altitudes, see Table VII. GLONASS and GalileoSat are Walker Delta Pattern constellations, whereas U.S. GPS is not.

The GPS constellations is made of 6 evenly distributed planes with four operational satellites each, GalileoSat has 3 orbital planes with 9 satellite each, and GLONASS has 3 orbital planes with 8 satellites each. The choice of trying to minimize the number of planes is motivated by the need to deploy more than one satellite with a single launch, thus minimizing the deployment costs. A different strategy has been outlined in section *Number of Launches and Constellation Deployment*, page 86-88.

Taking into consideration the requirements of symmetry, altitude and number of satellites of existing constellations for global navigation the following strategy for finding FCs suitable to the navigation problem has been developed.

The parameters entered by the user are:

N_s Number of satellites in the constellation.

$N_{d,\max}$ Maximum number of days for constellation periodicity.

$T_{\min}$ Minimum orbital period.

$T_{\max}$ Maximum orbital period.

$i_{\min}$ Minimum value of orbit inclination.

$i_{\max}$ Maximum value of orbit inclination.

N_{pts} Number of sample points to be used for orbit propagation.

These parameters are used to generate all possible FCs configurations, and these are saved in a suitable data structure, i.e. the F_c matrix, as explained before. Next all the configurations in which the satellites are not symmetrically distributed are eliminated, with the algorithm proposed in [30]. Finally all the remaining constellations are scanned to eliminate those configurations in which the satellites become very close for some time, i.e. configurations that are likely to provide a bad $GDOP$.

At this point the remaining configurations are candidate solutions for the Global Navigation problem. Typically, out of several thousands configurations, only a hundred remains at this step. For this reason an exhaustive search becomes possible and desirable.

The remaining configurations are examined to find those configurations that have the smaller $GDOP$ during the repetition time for receivers placed at each latitude (latitude is spanned by placing receivers from -80° to $+80^\circ$ latitude at 5° interval), i.e.

$$\min_i \max_t GDOP \tag{6.7}$$

The configurations under scrutiny are symmetric and uniformly distributed, thus the $GDOP$ characteristics repeat after a time T/N_s where T is the constellation repetition time: this symmetry is exploited to greatly reduce the amount of computations required.

Each configuration, being circular orbits and having all the integer parameters already set, must be only optimized for inclination. An inclination scan, of 1 degree, from -90 to +90 is performed for each configuration in order to find good values of the inclination that minimize the $\max GDOP$. During the inclination scan, if some configuration has a a value of the $\max GDOP < 6$, i.e. a promising constellation, a gradient search algorithm is used to find a more accurate value of the inclination that actually minimizes the $\max GDOP$ for that configuration. Thus this search technique is hybrid since it uses a linear scan to find a good initial guess for solving eq. 6.7.

Results

Fig. 35 show that although the mean $GDOP$ of both GPS and best FC (see Table VIII) are very similar, there is no marked improvement on the performance of the system overall. In some cases both the GPS and the best FC presents bad $GDOP$ configurations that yield poor position estimation.

Fig. 36 instead shows a comparison of the amount of time, relative to the time needed to complete the relative path in ECEF coordinates, in which the value of $GDOP$ of the best FC found with the algorithm described in this section and the GPS constellation. In this figure of merit the FC performs better than the nominal GPS system.

In summary this section showed that FCs can be used also in the domain of global navigation and outlined a method to find symmetrical FCs with configurations that have minimal risk of collision and therefore are likely to provide a good $GDOP$.

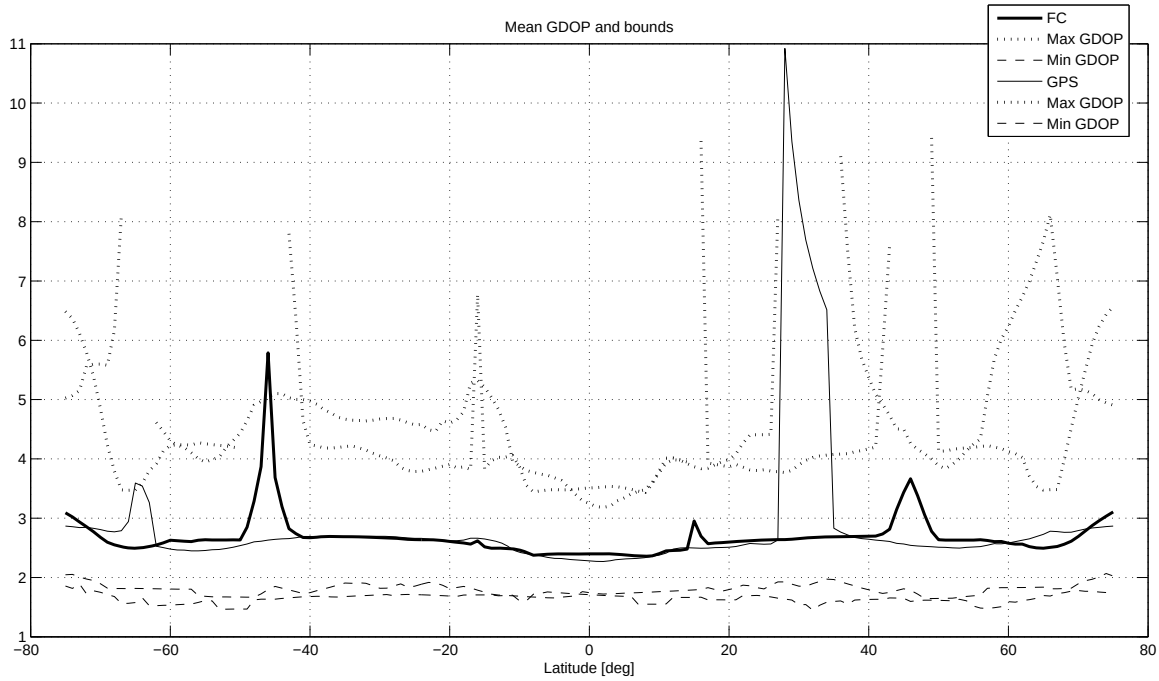


Fig. 35. Mean, minimum and maximum *GDOP* comparison of best FC configuration and GPS.

Table VIII. Best FC found for GDOP optimization with 24 satellites.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
23	9	24	7	8	5	16,180	108.6	*

D. Regional Coverage

Regional coverage is the problem of observing a specific Earth Region of Interest (ROI), either expressed with latitude and longitude limits, a physical feature (e.g. U.S. East coast), or political boundaries (e.g. Texas).

Regional coverage is particularly challenging to achieve with Walker Delta patterns, whereas FCs are ideally suited for the design of regional observation orbits

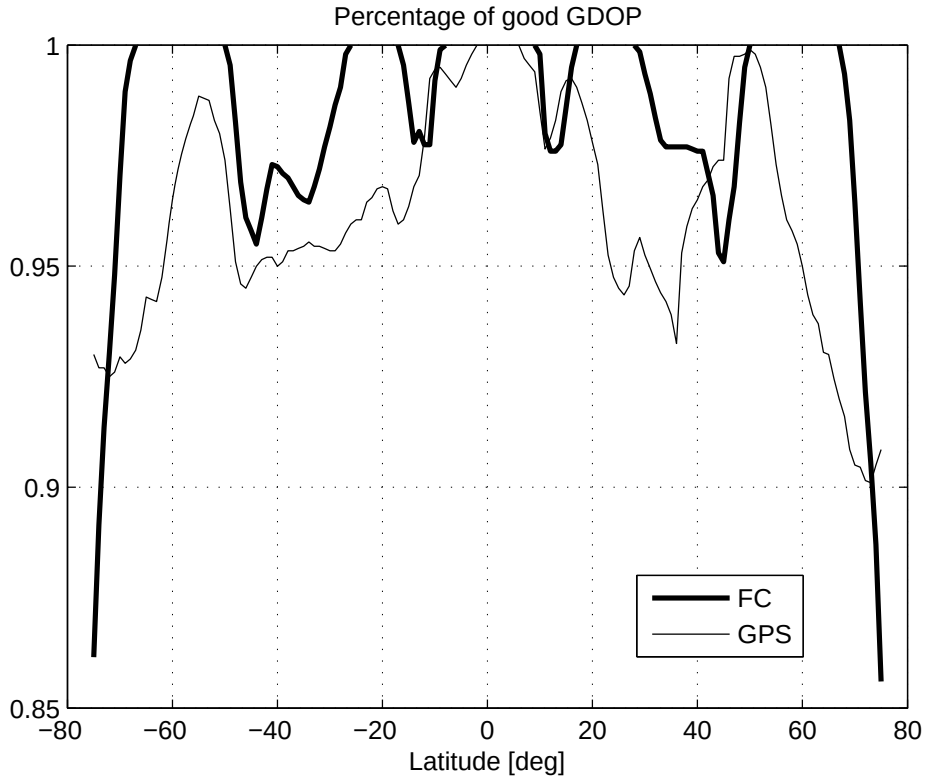


Fig. 36. Percentage of time for which $GDOP < 6$ comparison between best FC and GPS.

because they can be built using any compatible orbit and, more specifically, any *multi-stationary* orbit since they have a fixed path relative to the ECEF coordinates. The FC satellites can be uniformly distributed along the relative trajectory, and this satellite distribution is optimal for persistent observation over the region. In other words, the time interval between subsequent satellite observation is constant, while Walker constellations provide different time interval gaps, sometimes overlapping.

As an example, the ROI has been chosen between 73.5°E and 135.5°E in longitude and between 18°N and 53.5°N of latitude, i.e. approximately above China, Taiwan, and Mongolia. Two distinct FC have been designed. The first one uses 4 satellites to provide persistent coverage with at least one satellite using an $N_p = 3$, $N_d = 1$ compatible orbit (Broglio's orbit) while the second one uses 5 satellites to provide persistent coverage with at least two satellites using an $N_p = 2$, $N_d = 1$ compatible orbit (Molniya multi-stationary orbit). All the orbits are critically inclined to avoid expensive perigee maintenance.

Example 1 - Broglio-Type Flower Constellation

The FC configuration chosen as an example for this problem is a 4 satellite, critically inclined, Broglio-type that has been optimized to continuously observe any point of the ROI with a minimum of 1 satellite.

The chosen configuration is symmetrical and, in ECEF coordinates, has three petals ($Np = 3$) distributed 120° apart from each other. The petals correspond to apogees where the satellites will dwell most of the time; the ground track plot is shown in Fig. 37, and the corresponding FC parameters are shown in Table IX. Hence the optimization process consists in finding the value of Ω_0 that maximize coverage over the ROI. One possible solution consists of placing one petal directly above the ROI, but considering that sensors can also be made to look sideways, the actual range of

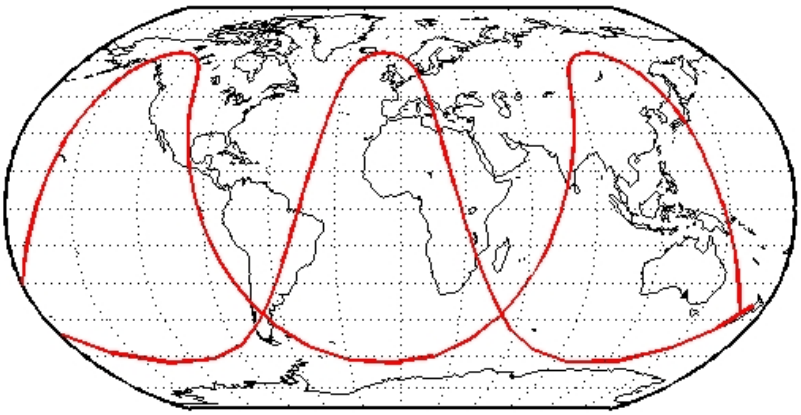


Fig. 37. Ground track of Broglio FC, optimized to provide maximum coverage above the ROI (China).

Table IX. Broglio-type Flower Constellation for regional coverage parameters.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
3	1	4	1	4	0	500	63.4	270

good solutions is increased.

Given the limited size of the search space, Ω_0 in fact belongs to the interval $[0°, 120°]$ because of symmetry, an exhaustive search can be performed by sampling the interval with a step of desired resolution (e.g. a fraction of a degree) and use an exhaustive search to find the value that offers the best coverage. The resulting orbital elements of the four satellites are shown in Table X.

Table X. Broglio-type Flower Constellation.

Satellite	a [km]	e	i [deg]	ω [deg]	Ω [deg]	M_0 [deg]
A1	20270.42	0.6607	63.4	270.0	80.8	0.0
B1	20270.42	0.6607	63.4	270.0	170.8	90.0
C1	20270.42	0.6607	63.4	270.0	260.8	180.0
D1	20270.42	0.6607	63.4	270.0	350.8	270.0

In order to compute the coverage characteristics over the ROI, China in this example, a number of virtual ground stations have been uniformly distributed over the ROI. For each site, and for the duration of the simulation, the coverage characteristics of each virtual ground station have been computed and stored in a visibility matrix M whose rows represents the visibility history on the i-th ground site and the columns represent the j-th time interval of the simulation. Therefore each element $M(i, j)$ contains the number of satellites visible from ground site i at time t_j.

A similar interpretation can be applied to Fig. 38 which also shows how the percentage of visibility of the region is influenced by the elevation angle: the lower the minimum admissible elevation angle, the higher the coverage percentage is. The field of view of the sensor, i.e. a cone of angle η, can be much higher than the physical field of view of the instrument if the sensor is actuated to look off nadir, as it is done for scanning sensors available to modern reconnaissance missions.

Fig. 39 shows that $\varepsilon = 10^\circ$ guarantees continuous 100% coverage of the whole ROI, whereas with $\varepsilon = 20^\circ$ has limited coverage ($65^\circ < \Omega_0 < 90^\circ$), and with $\varepsilon = 30^\circ$ there is an optimal orientation for the FC associated with $\Omega_0 \simeq 82^\circ$. Fig. 40 shows the observation matrix M for the optimal $\varepsilon = 30^\circ$ solution: 21 sites (out of 22) have 100% persistence coverage and one site has its visibility reduced to 75%.

Example 2 - Molniya Type Flower Constellation

In the second example a 5 satellite, critically inclined, Molniya-type (see Table XI) FC has been adopted to provide continuous observation of the ROI with a constrained persistence of at least 2 satellites. As for the previous example, the limited range of the Ω_0 parameter to be optimized allow for an exhaustive search.

The corresponding satellite orbital elements are shown in Table XII. The level of coverage obtained by varying Ω_0 is shown in Fig. 42 for 3 different values of ε: 25°,

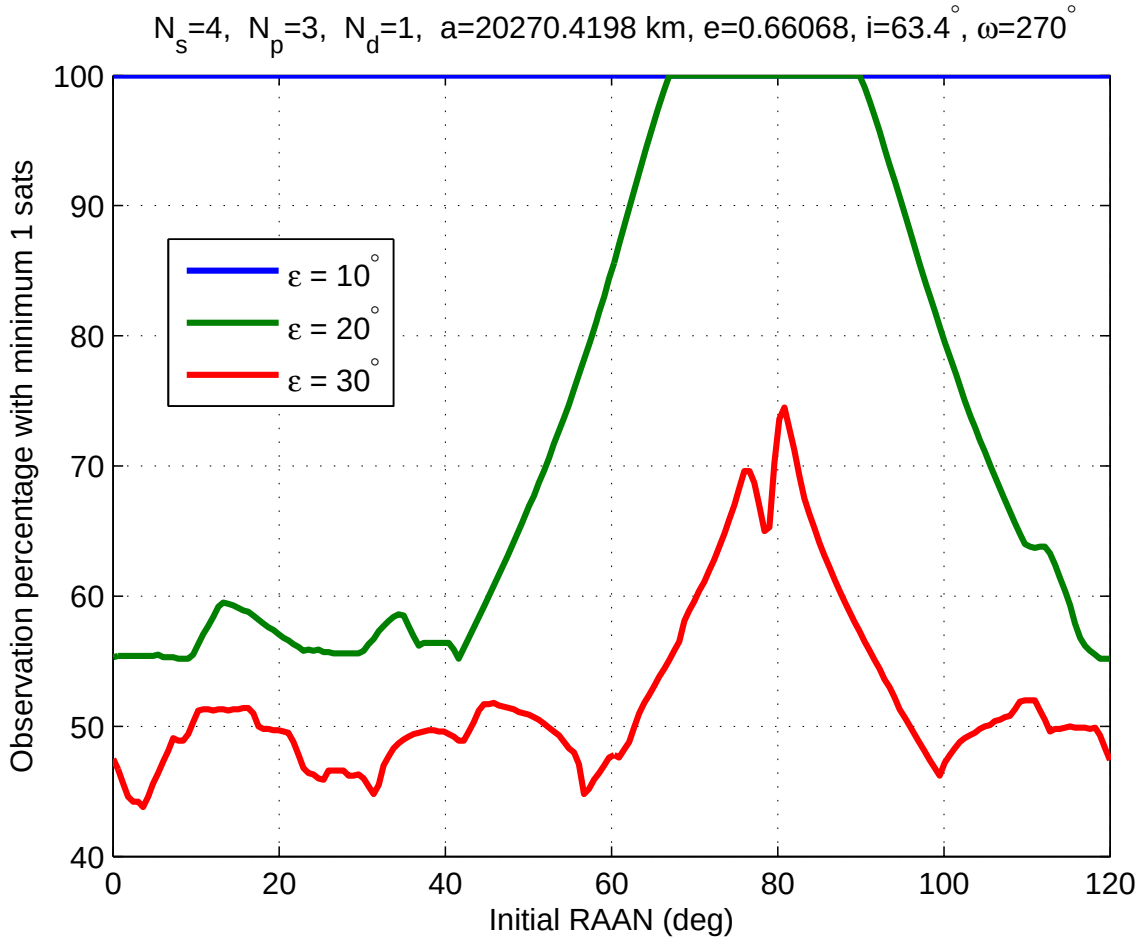


Fig. 38. Broglio-type FC: percentage of visibility as a function of Ω_0, for different values of the minimum grazing angle, ε.

Table XI. Parameters of Molniya-type Flower Constellation for regional coverage.

N_p	N_d	N_s	F_n	F_d	F_h	h_p [km]	i [deg]	ω [deg]
2	1	5	1	5	0	500	63.4	270

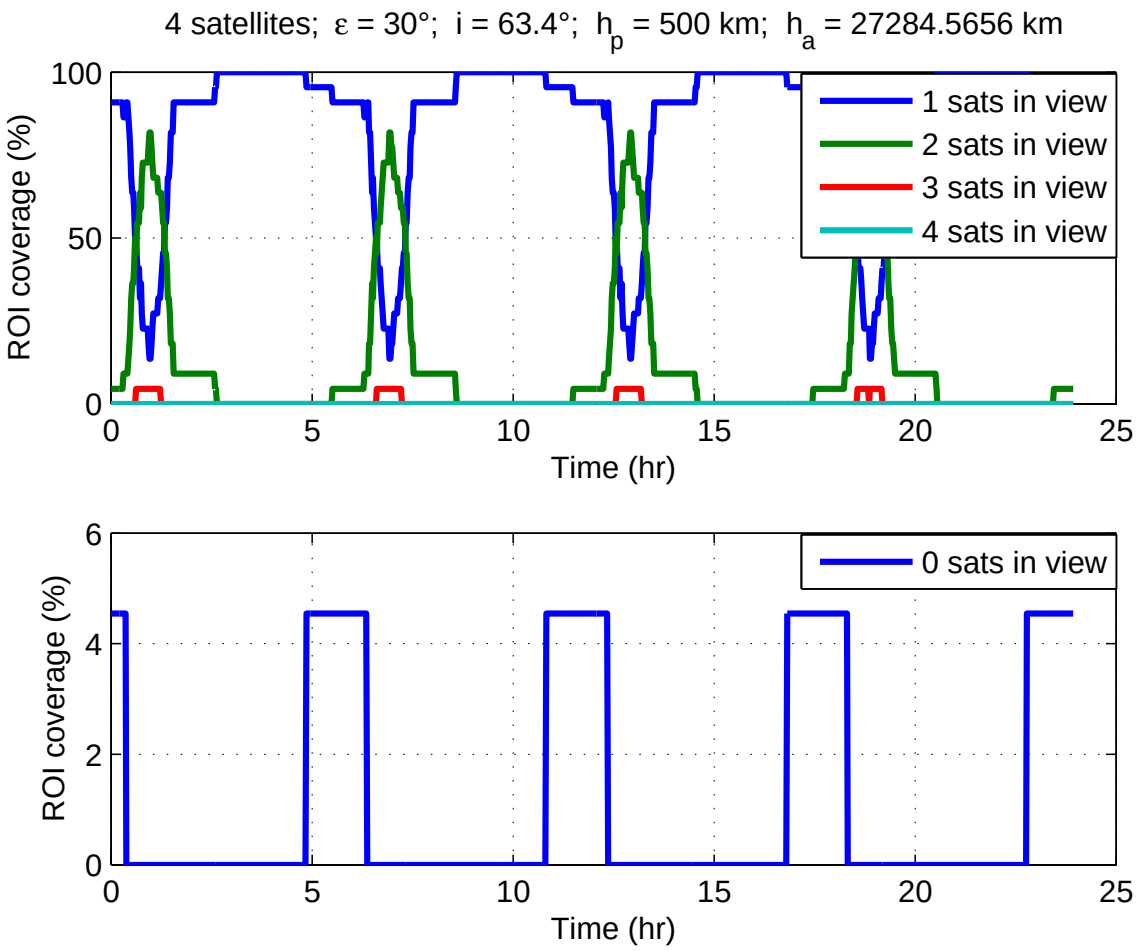


Fig. 39. Broglio-type FC: detailed visibility results for $\varepsilon = 30°$. Broglio-type FC.

Table XII. Molniya-type FC: satellite orbital elements.

Satellite	a [km]	e	i [deg]	ω [deg]	Ω [deg]	M_0 [deg]
A1	26561.8	0.741	63.4	270.0	107.64	0.0
B1	26561.8	0.741	63.4	270.0	179.64	216.0
C1	26561.8	0.741	63.4	270.0	251.64	72.0
D1	26561.8	0.741	63.4	270.0	323.64	288.0
E1	26561.8	0.741	63.4	270.0	35.64	144.0

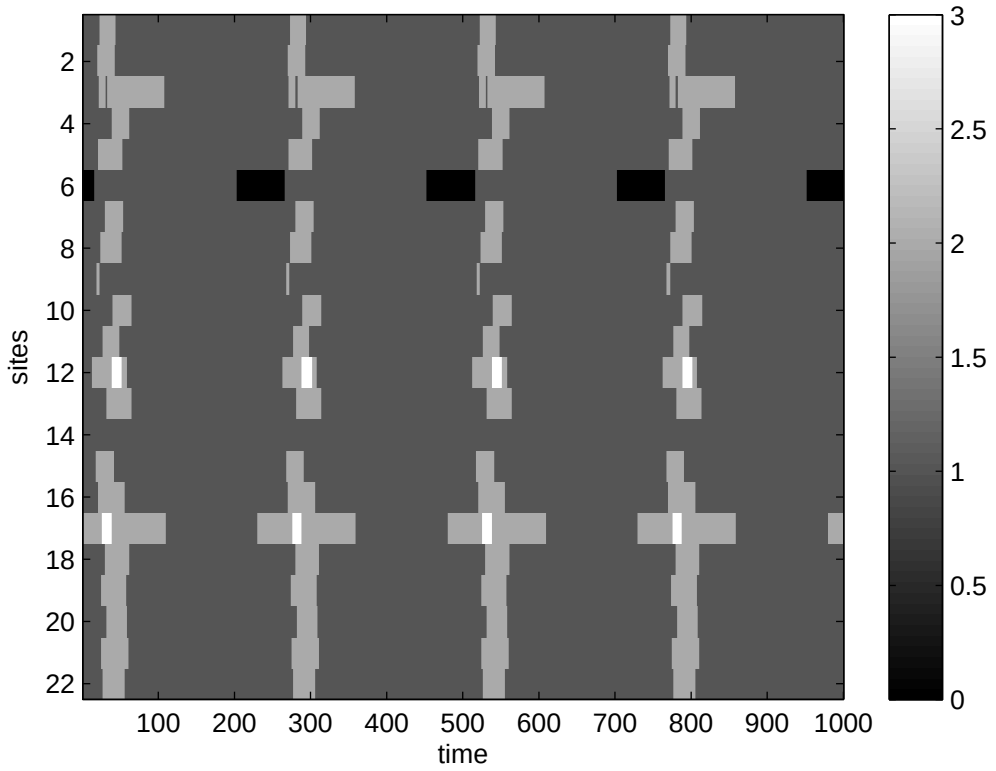


Fig. 40. Broglio-type FC: number of satellites visible from each ground site as function of time; the legend on the right relates the number of visible satellites and color. The vertical axis is the site number and the horizontal axis is the time interval.

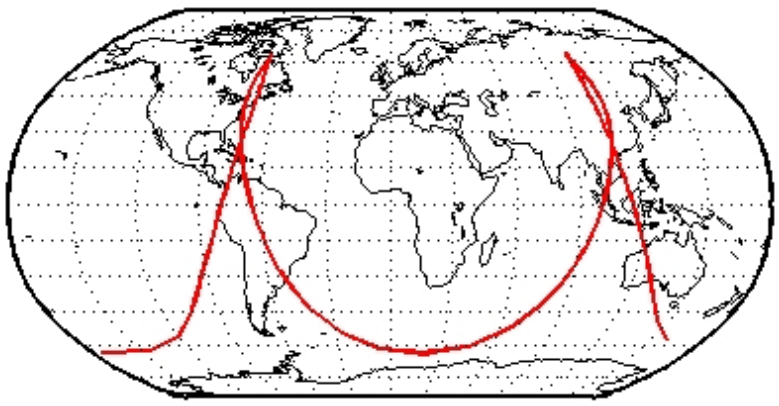


Fig. 41. Ground track of Molniya-type FC, optimized to provide maximum coverage above the ROI (China).

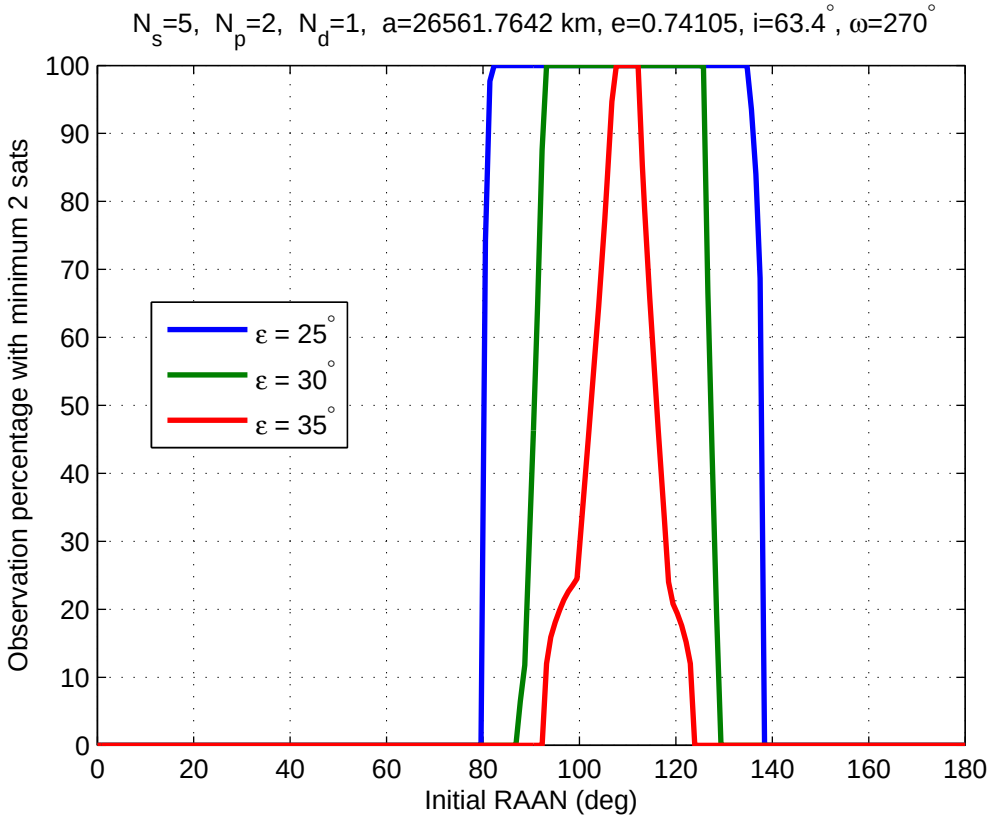


Fig. 42. Molniya-type FC: Percentage of coverage as a function of Ω_0, for three values of the grazing angle ε.

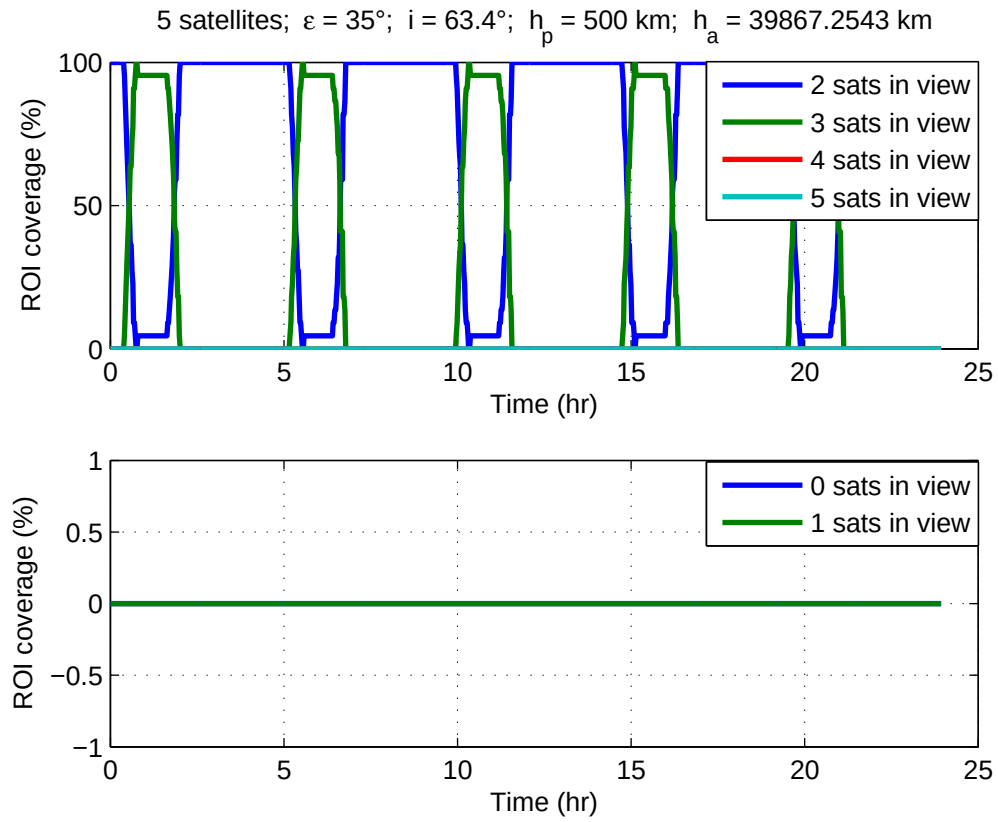


Fig. 43. Molniya-type FC: detailed visibility results for $\varepsilon = 35°$ case.

30°, and 35°. In this case the best solution corresponds indeed to placing the petals above the ROI, since there are only two petals, 180° apart, as shown in Fig. 41.

Similarly as the previous example, Figs. 43 and 44 show the history of the number of visible satellites for each ground station. From the latter figure it is clear that at least one satellite is always in view of all the ground sites.

The two examples illustrated above show the expected trade off between altitude and coverage, but it must be emphasized that the second example had one more satellite available to improve the coverage characteristics. These examples also serve the purpose of illustrating how the FC can be utilized to propose novel configurations (example 1) and still be competitive with well known solutions (example 2). Finally

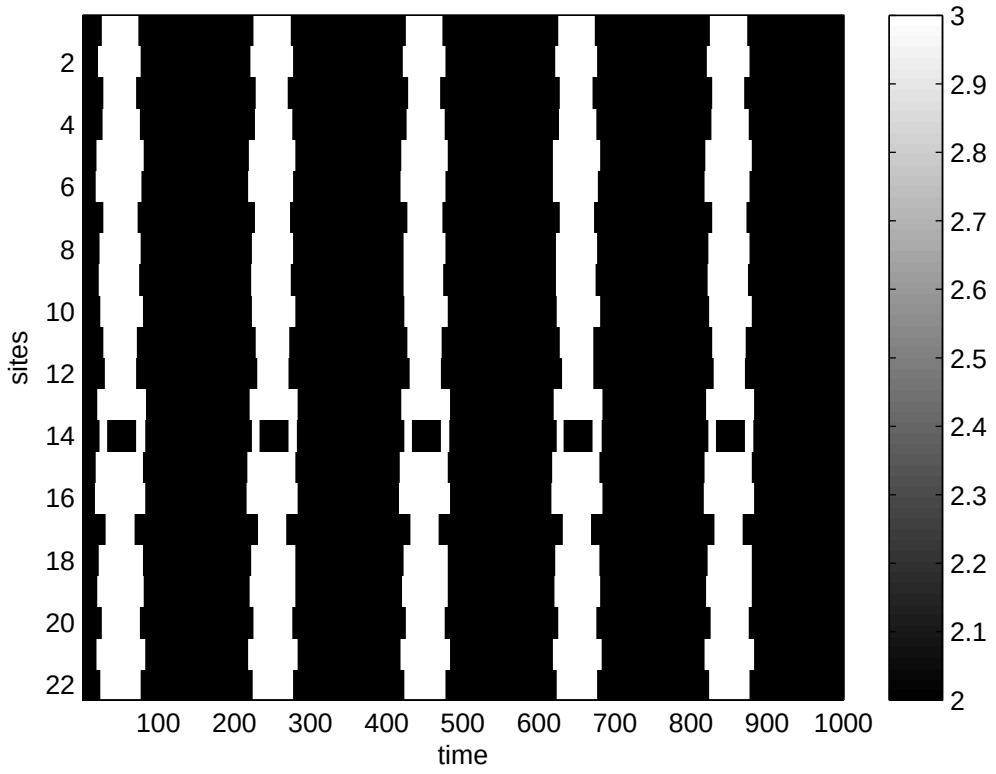


Fig. 44. Molniya-type FC: number of visibile satellites for each site as function of time, for the $\varepsilon = 30^\circ$ case.

this last example demonstrates also how FCs can be used to reproduce already known types of constellation very easily.

E. Reconnaissance Missions

Reconnaissance from space serves, as it historically always did, an important military role. The first examples of aerial observation for military intelligence gathering date back, in fact, to the American Civil War. Both sides used aerial balloons to gather information on enemy positions and movement of troops. The fact that the balloon had to be constrained with a rope to the ground limited altitude and range, thus they were confined to tactical use only (i.e. local on the battlefield).

The first use of airplanes in World War I was also for reconnaissance of enemy movements, artillery positions, and offensive/defensive readiness assessment. The much better range and freedom of airplanes allowed for a shift from tactical to strategic reconnaissance. Reconnaissance planes played an important role in World War II[11], and even more on the first decade of the Cold War but then, with the increase in range and effectiveness of Surface to Air Missiles (SAMs) the role of spy planes was sensibly reduced. When the USSR satellite *Sputnik* (eng. literally "Satellite"), October 1st 1957, started the space age, it became quickly clear that the next frontier for strategic intelligence gathering was space.

The *Corona* program from, 1960 to 1972, was the U.S. first space-reconnaissance program, at the time top secret, used to assess Soviet weapon systems and operational capabilities. The program was very expensive for the time, but it was later said that the information gathered through the *Corona* satellites was worth ten times as much in terms of the impact it had on strategic decision making.

[11]See the effect of long range reconnaissance in the Pacific theater, during the battle of Midway, June 7th 1942

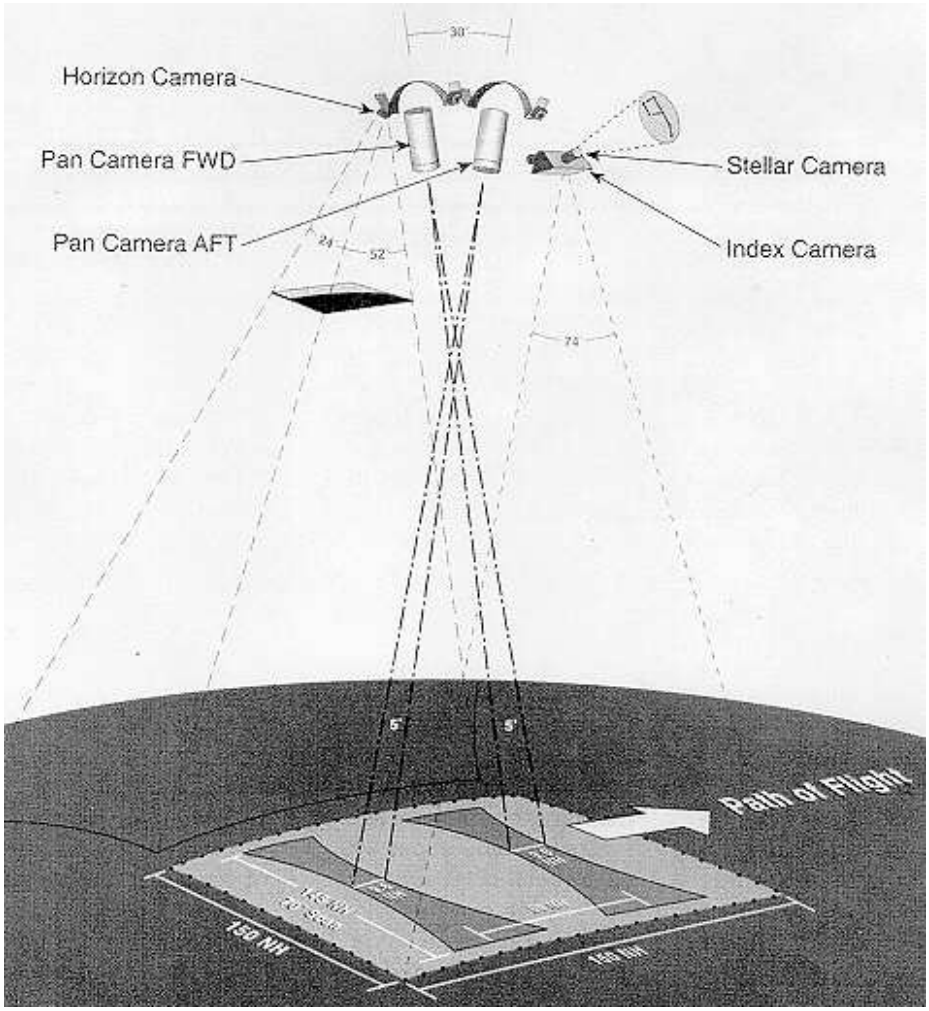


Fig. 45. *Corona* camera system and observation geometry [67].

Sensor Types and Applications

The *Corona* satellites offered panchromatic (i.e. black and white) images at a resolution of 10 m, which progressively improved during the years up to 2 m. It was also equipped with counter-rotating stereo cameras to acquire 3D images of the terrain: this was the first time it was ever tried from space. The film was dropped and re-entered in the atmosphere with a capsule which was recovered in mid-air during the parachute descent phase [66], a diagram of the camera system is shown in Fig. 45 [67].

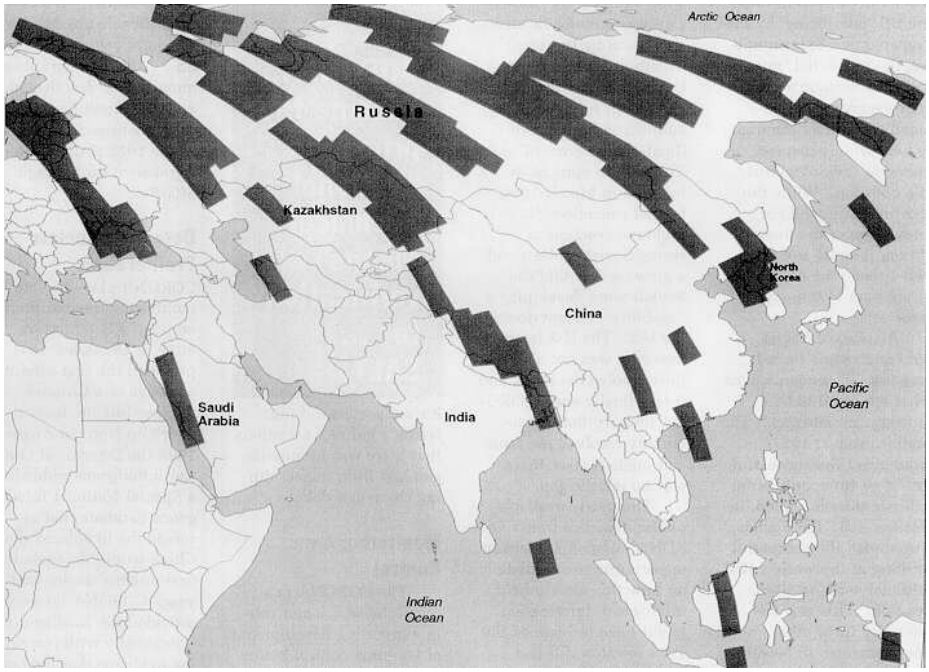


Fig. 46. *Corona* 4 day coverage [68].

The *Corona* spacecraft orbits were particularly low, i.e. below 185 Km average perigee, with an inclination of 60° to 110°, and slightly eccentric (average apogee at 278 Km, thus $e = 0.007$); the low altitude limited their life to only 19 days because of atmospheric drag. This fact is important to point out: the mission life of such satellites was dramatically short, and they were very expensive to build and put in orbit too; nonetheless all these considerations were deemed secondary with respect to the importance of the mission accomplished. Figs. 46 [68] and 47 [69] show the ground coverage obtained by the spacecraft in 4 days and an example of the resolution that could be obtained in 1967, respectively.

The use of elliptical orbits is sometimes criticized because these make the mission design and the requirements on spacecraft attitude more complicated. Furthermore, if not critically inclined, the perigee maintenance cost for non-circular orbits may quickly become prohibitive for long term missions. In the near future however the

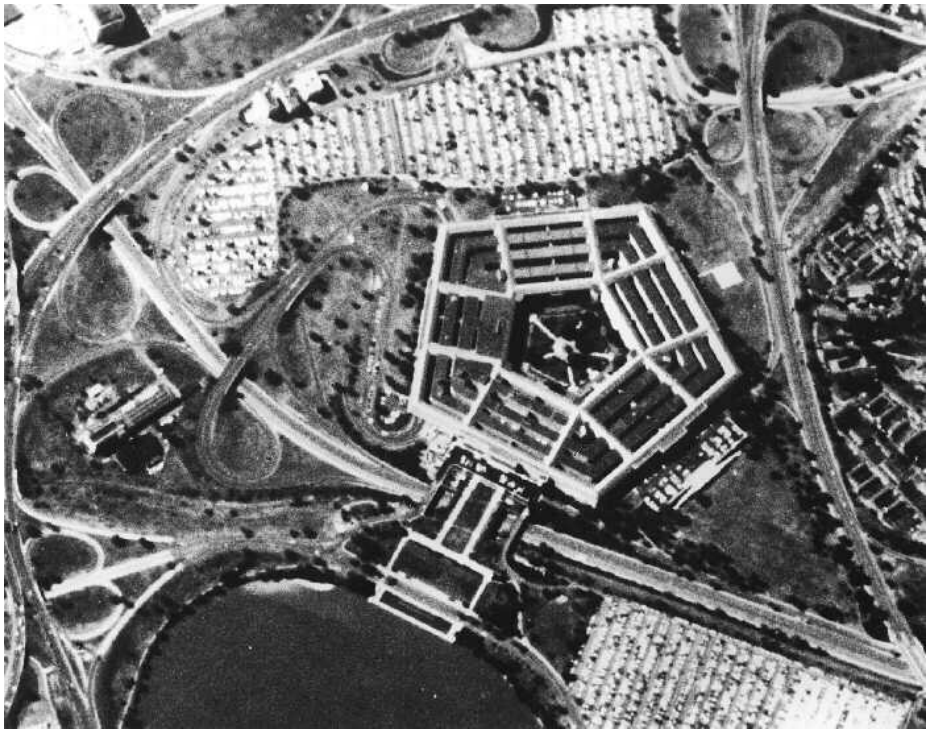


Fig. 47. Image of Pentagon, Washington DC taken by *Corona* Spacecraft, 25 Sep 1967 [69].

possibility of launching and keeping in orbit a constellation of micro-satellites for a limited period of time might be considered a good investment if the mission that such constellation accomplishes is worth it.

In the present days the capabilities, and the range of sensor technologies, available for space-reconnaissance have dramatically improved. Panchromatic and color images are available presently at a resolution of 1 m, soon to 0.5 m for civil and commercial applications. The resolution limit of 0.5 m is not a technological limit: *IKONOS* corporation claims that 0.4 meters is possible with their new cameras but for commercial applications they are not legally allowed to release images of better resolution than 0.5 m. It can be assumed then that military cameras and sensors are probably capable of even better resolution.

Images from IKONOS, and from its predecessor, *Quickbird*, have provided the basis for the commercial success of mapping systems integrated with GPS navigation. The coupling of technologies like wireless communications, global navigation, and networking is going to have a profound impact in terms of commercial innovation and creation of new markets in the immediate and near future. Integration of technologies like *Google Earth*, smart phones, GPS navigation, and *web 2.0* is quickly becoming a pervasive reality.

Multi-spectral and hyper-spectral imaging can provide information on non-visible wavelengths in addition the the visible wavelengths, thus enhancing considerably the range of possible applications. Multi-spectral imaging adds the capability to see mainly into the infrared (IR). One reason to do this is the need to see through the cloud cover that often limits the usefulness of visible imaging; an application of paramount importance is detection of long range or ICBM missile launches for early warning and missile defense systems.

The Landsat program (Earth-Resource Technology Satellites, NASA) used a

multi spectral imager together with a panchromatic TV system capable of a 15 m resolution; it provided imagery in 7 spectral bands (3 Visible, 1 Near Infrared (NIR), 2 Short Waves Infrared (SWIR), and 1 thermal) as shown in Table XIII [66]. Landsat

Table XIII. Landsat multi-spectral bands.

Type	*Wavelength* (nm)	*Resolution* (m)
Visible	450-520	30
Visible	520-600	30
Visible	630-690	30
NIR	760-900	30
SWIR	1550-1750	30
Thermal	10.4 μm - 12.5 μm	120 (60)
SWIR	2090-2350	30

satellites were placed in circular, sun-synchronous ($i = 97°$) LEO orbit, raised to 750 Km of altitude for later spacecrafts. The revisit time for a generic site was of 16 days. Such characteristics can be easily reproduced with a flower constellation.

Hyper-spectral imaging extends the concept of multi-spectral: images are provided on a continuous spectrum, rather than a disjoint set of bands as in multi-Spectral imaging. Multi-spectral and hyper-spectral imaging are primarily useful to identify and classify materials present in the image. Thus it is possible to distinguish between different kinds of crops, the state of health of vegetation, type of soil and, at higher resolution, it is even possible to distinguish between camouflaged man made objects and natural terrain which would be very difficult to discern otherwise.

Synthetic Aperture Radar (SAR) extends even further the range of remote sensing capabilities: unlike the passive imaging techniques summarized so far it is active, i.e. the target is illuminated by the radar itself, and provides all-weather imaging

capabilities, independently of the sun. It can even penetrate for a small depth the Earth surface and therefore it is capable of detecting buried objects.

Images obtained with radar at different frequencies and polarizations can be combined to yield an image in which materials of interest are highlighted. This process requires substantial tests and tuning before being effective, but the results can be dramatic. SAR from space has been demonstrated by the Shuttle Endeavor in 1994; an image of mount Etna, Sicily, obtained with the SIR-C/X-SAR instrument during the same mission is shown in Fig. 48 [70]: note how lava flows of different eruptions appear in different colors in this synthetic image. Roughness is also an important factor in backscattering (i.e. the scattering of the signal reflected by the surface toward the receiver).

Fig. 48. SIR-C/X-SAR Image of mount Etna, October 11, 1994. The imaged area is 51.2 km by 22.6 km [70].

Optimization for the Reconnaissance Problem

Three programs have been developed to explore the possibilities of using natural orbits in the reconnaissance problem.

- **Program GroundTargets1**. In the first program a spacecraft must survey a set of ground targets. The objective is to find the orbit that will allow the spacecraft to visit the ground sites within a specified amount of time, or that will bring the spacecraft within a minimum required range from each site to obtain the desired imaging resolution. This part of the work builds from the results presented in [71] and it is mostly a review and better integration of the code in the current framework, therefore for details and results refer to the cited work.

- **Program GroundTargets2**. This program is similar to the previous, GroundTargets1: the task is simply that of finding the optimal orbital parameters to accomplish the reconnaissance mission by maximizing the dwell time. The difference between this program and the first is that the first program does not try directly to optimize the dwell time, but simply to find an orbit that will visit all the ground targets within a specified amount of time. By using efficient coding and propagation routines, this new program performs a direct optimization of the dwell time over the ground targets.

- **Program J2Exploitation**. In this program the spacecraft is already in orbit performing a reconnaissance mission and the set of ground sites is altered (i.e. some site is added or removed, or the relative importance of the site is changed). The goal of the optimization is finding the optimal orbit that allows the reconnaissance of the ground sites maximizing the dwell time by using no more than the specified amount of fuel for the impulsive maneuver required to adapt the orbit to the updated mission requirements. Since there is no time constraints, the effect of the $J2$ perturbation (Ω drifting) can be exploited to achieve the desired orbit reducing the fuel budget.

The final desired output is a compatible orbit that maximizes the dwell time of the spacecraft on the ground sites according to the situation outlined in the three scenarios. Using a compatible orbit ensures that the revisiting time will be within the desired limits. Furthermore, if such orbit is found, it is immediately possible to design a FCs with the assigned number of satellites that will provide the desired coverage over the ground targets.

Program GroundTargets2

This is similar to the first in purpose as it is shown in Fig. 49, but provides a different implementation. GroundTargets1 uses the visiting time of each site as genes in the chromosome; this avoids the need for propagation, but becomes less effective as the number of sites increase. For this reason a solution that use some form of efficient propagation, but is more scalable has been sought.

Performing orbit propagation implies the evaluation of orbital position vector $\vec{r}$ from time. The slowest part of the procedure is represented by solving Kepler's equation: in order to make it more efficient, fast KE solvers, routines `kepler_danby.m` and `kepler_danby_mikkola.m`, were optimized for Matlab execution.

Program `GroundTargets2.m` uses the chromosome structure shown in Fig. 50. The GA proposes the default range shown in Fig. 51 for variables in the chromosome.

These ranges can be modified by the user. For instance, the user can select `ECC_max = 0` if he wants to see the optimal circular orbits, or `INC_min = INC_max = 63.4` if only critical inclined orbits should be considered, and so on. The reason why we selected these default values are listed in the following:

- `ECC_max` (highest value for orbit eccentricity) is associated with minimum perigee altitude;

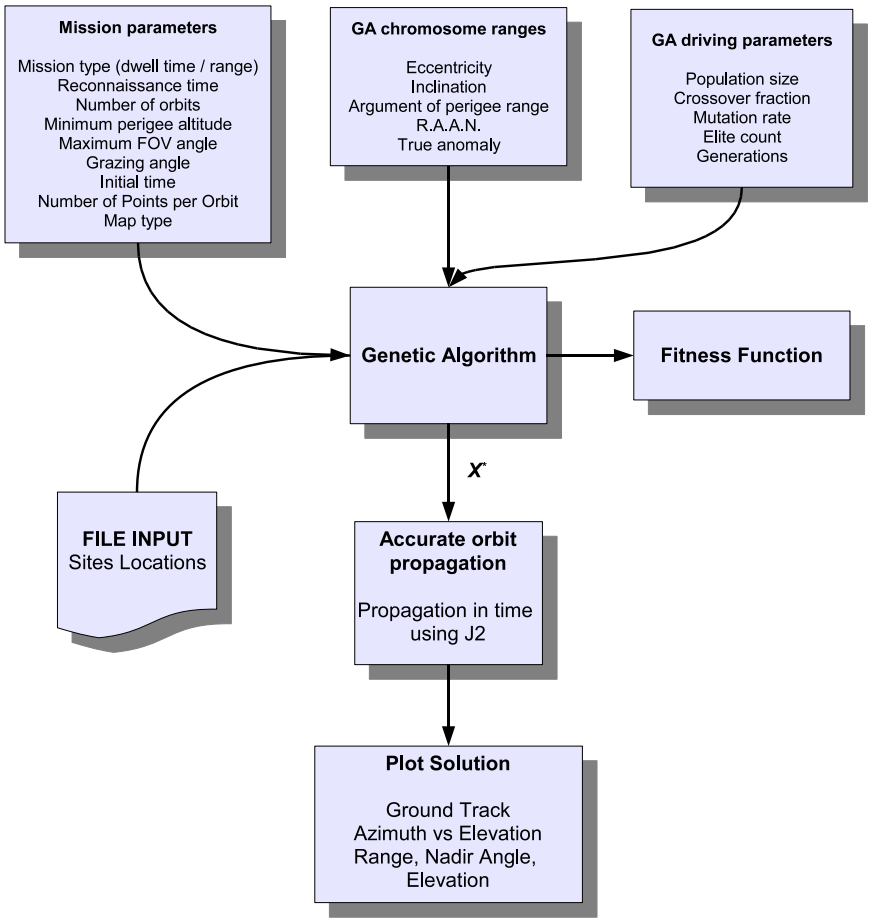


Fig. 49. Functional diagram of program GroundTargets2.

- `INC_min` (minimum inclination) has been selected equal to the latitude of the site with highest latitude.
- `INC_max` (maximum inclination)
- `RAAN_max` (maximum RAAN) has been selected because compatible orbits with Ω and $\Omega + k2\pi/N_p$ have same ground tracks.

The `GroundTargets2.m` program prompts the sequence of dialog windows shown in Figs. 52(a), 52(b), 53(a), and 53(b), allowing the user to select and modify default input parameters.

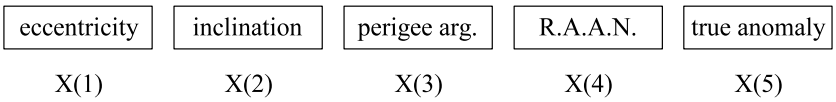
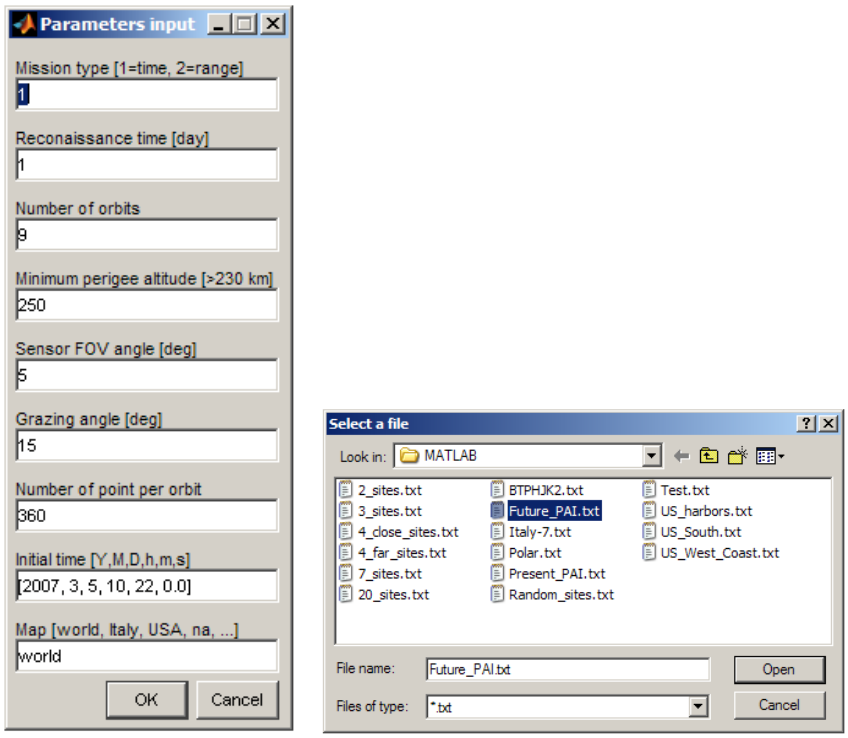


Fig. 50. GroundTargets2 program chromosome.

```
ECC_min  = 0;
ECC_max  = 1 - (EarthRadius + hp_min)/sma0;
INC_min  = cw*max(abs(Sites_Latitudes));
INC_max  = 180 - INC_min;
ARP_min  = 0;
ARP_max  = 360;
RAAN_min = 0;
RAAN_max = 360/Np;
TA0_min  = 0;
TA0_max  = 360;
```

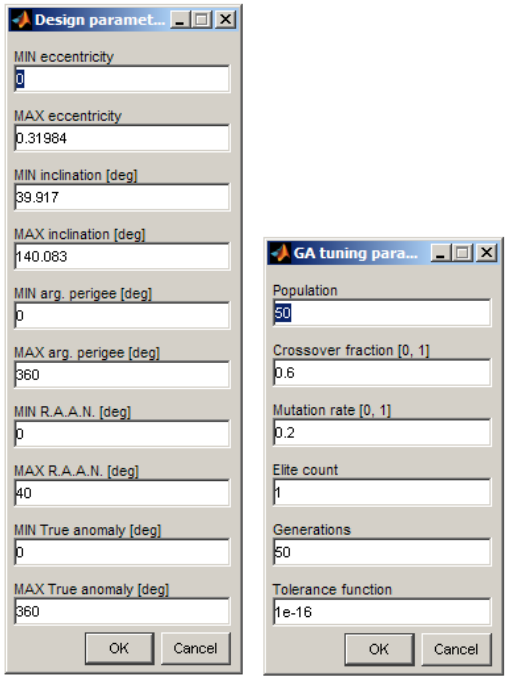
Fig. 51. Default values assigned to ranges of variables (genes) in the chromosome.

The console output is provided in Figs. 54, 55 and 56. The `GroundTargets2.m` program provides in outputs the plots in Figs. 57, 58, and 59.



(a) Mission input. (b) Input file selection.

Fig. 52. GroundTargets2 input dialogs.



(a) Gene ranges. (b) GA tuning.

Fig. 53. GroundTargets2 program input dialogs.

```
Optimization terminated: maximum number of generations exceeded.


====>   Mission parameters  <=====

    Mission type = 1

Mission duration = 1 days

Number of orbits = 8

   Cost function = -603.6063
```

Fig. 54. Program GroundTargets2 output summarizing mission parameters.

```
====>   Orbit Parameters  <=====

  time = 05-Mar-2007 10:22:00 (JD=2454164.9319)

     T = 2.9828      hr

   sma = 10519.8382  km

   ecc = 0.37078

    hp = 241.2041    km

    ha = 8042.1984   km

   inc = 63.4958    deg

 omega = 16.6815    deg

  RAAN = 27.2728    deg

TrueAn = 204.9205   deg

MeanAn = 228.6515   deg

     x =-7229.9134   km

     y =-8284.3678   km

     z =-8122.5284   km

    Vx = 3.8166  km/sec

    Vy = 0.66109 km/sec

    Vz =-2.3287  km/sec
```

Fig. 55. Program GroundTargets2 output of orbital parameters.

```
====>   Perigee Maintenance Cost  <=====

     Delta v = 0.077243 km/sec/yr

100*(1-m/m0) = 0.0031508 %/yr [Isp = 250 sec]


Site: Brazil observed for 0.21674 hr (w = 0.33333)

Site: Italy observed for 0.058354 hr (w = 0.33333)

Site: Australia observed for 0.22508 hr (w = 0.33333)
```

Fig. 56. Program GroundTargets2 output after optimization.

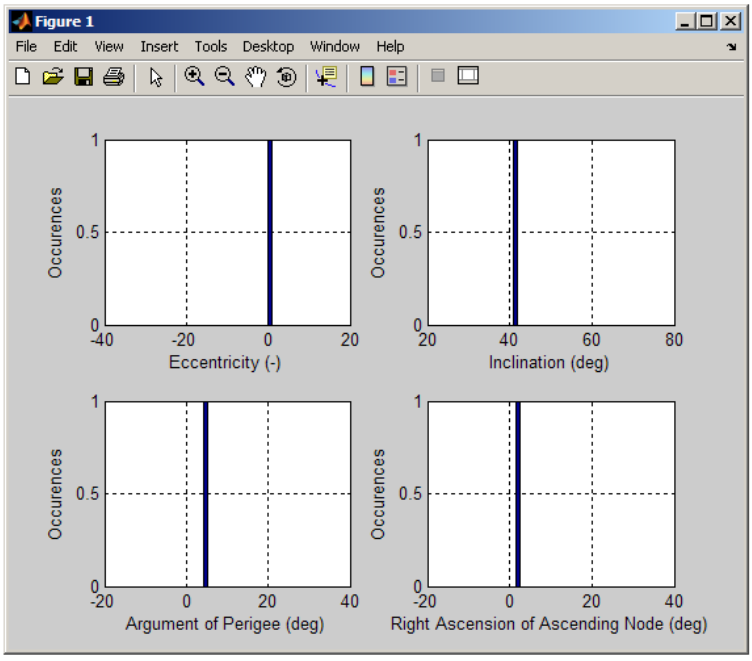


Fig. 57. GroundTargets2 output: Statistics of last generation.

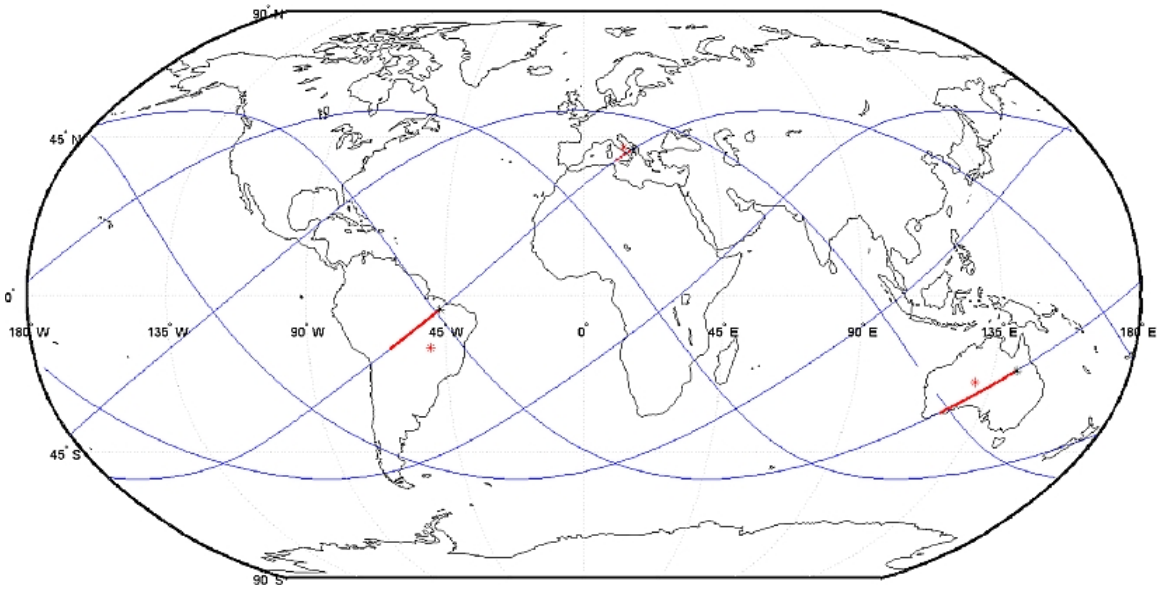


Fig. 58. GroundTargets2 output: Ground track and observations.

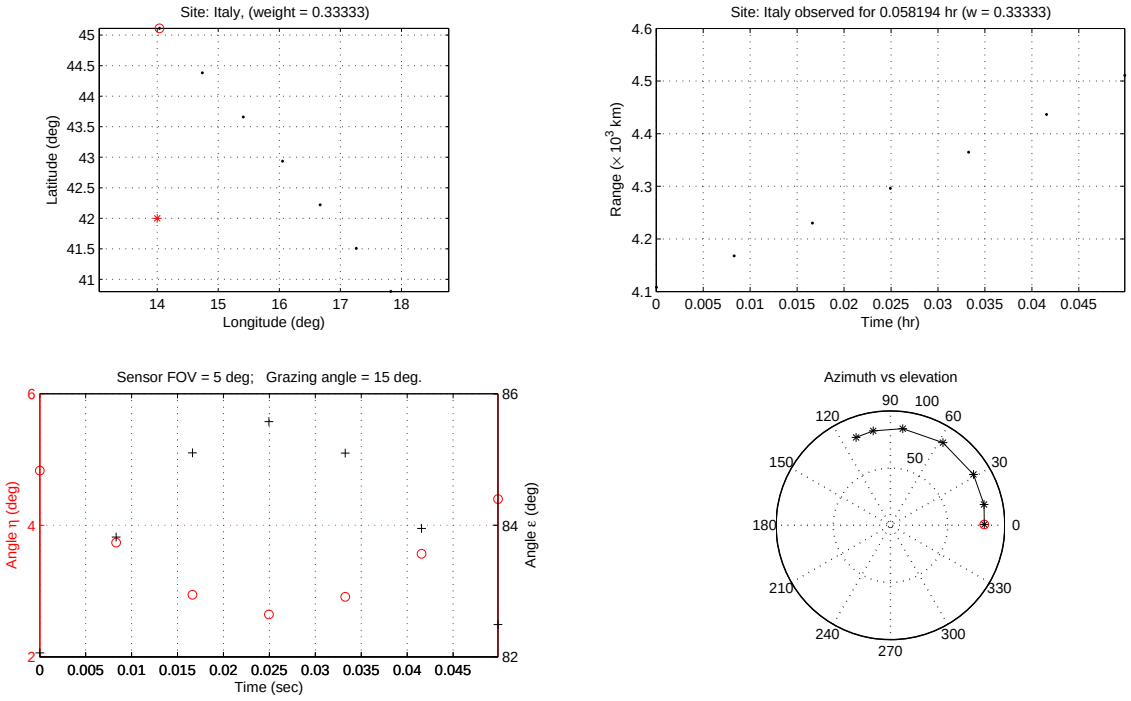


Fig. 59. GroundTargets2 output: Site observation characteristics.

Program J2Exploitation

Program `J2Exploitation` considers the scenario where a reconnaissance satellite is already in orbit with assigned initial orbital elements $[a_0, e_0, i_0, \omega_0, \Omega_0, \varphi_0]$, specified for an initial time, T_0. This spacecraft has a limited Δv_{tot} budget to observe a new (or a modified) set of N Earth sites, each one with its own importance (relative weight, w_i). The observation mission must be completed within a limited time T_f.

The purpose is to find when, within the time limit T_f, a single impulsive maneuver can be applied to move into a new orbit optimized for the updated ground targets. The program uses the four-element chromosome structure shown in Fig. 60.

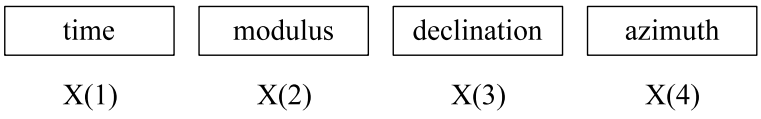


Fig. 60. J_2 Exploitation Chromosome.

The functional diagram of the program is shown in Fig. 61.

This new approach is NOT constrained by the adoption of using compatible orbits and the orbit propagation considers the J_2 perturbations. In particular, this approach allows us to investigate how to use orbit perturbations to reduce the fuel budget required to meet the update mission requirements.

The idea leading to this approach is originated by the assumption of having (in the near future) available on-orbit spacecrafts that are already equipped with specific instruments to perform effective reconnaissance missions. Therefore scenario considers a reconnaissance spacecraft already in space and with available fuel for changing its orbit with a single low-cost impulsive maneuver. This maneuver should be executed within a reconnaissance limit time T_f while the maneuver itself is performed

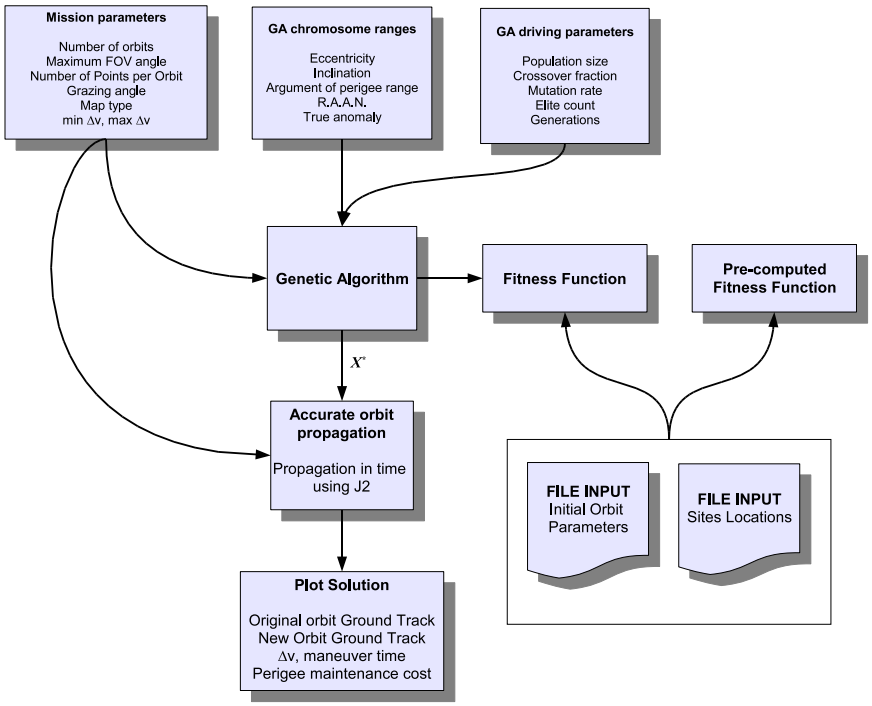


Fig. 61. Functional diagram of program J2Exploitation.

at time T_m.

As in the approach that uses compatible orbits, the "J_2 Exploitation" approach also finds the optimal solution using Genetic Algorithm. In this case, however, the reconnaissance problem is split in two orbital trajectories:

- First part, from T_0 to T_m, where the spacecraft is on its initial reconnaissance orbit
- Second part, from T_m to T_f, separated from the first by a low-cost maneuver.

Under this assumption, the design parameters consist of the maneuver time event (T_m) when to move into a "close orbit" and the maneuver vector ($\Delta\vec{v}$).

The main idea of the "J_2 Exploitation" approach is hereafter summarized. Initial data are:

- Spacecraft status
  - Position,
  - Velocity,
  - Initial GMT time ($T_0 = 0$),
- Earth sites data (they are n)
  - Latitude,
  - Longitude,
  - Relative weight (weights are normalized, e.g., their sum is equal to one).
- Reconnaissance time limit, T3, (since T_0).

Optimization constraint parameters:

- Maneuver time range, $[T_1, T_2]$, where $0 = T_0 \leq T_1 \leq T_2 \leq T_3$.
- Impulse range, $[\Delta v_1, \Delta v_2]$, where $0 \leq \Delta v_1 \leq \Delta v_2 \leq \Delta v_{\max}$.
- Optimality cost function type [dwell time or resolution]

The program `J2Exploitation` reads the following two input text files:

1. *Earth site input file.* The filename of this input file can be anyone except `Random_sites` (see later for the specific meaning of this file), but the filetype (file extension) must be `.txt`. An example of an Earth site input file is the following, containing some USA harbors. The number of Earth sites must be limited according to the amount of computation resources available, or the computation time becomes too long. Each line contains the information of a single Earth site. The information must be provided according to the following

Table XIV. Sites used by the J2Exploitation program.

Longitude	Latitude	Weight	Location
-76.61	39.29	1.0	Baltimore
-71.08	42.35	1.0	Boston
-67.00	44.90	1.0	Eastport
-70.25	43.67	1.0	Portland
-71.40	41.83	1.0	Providence
-82.55	27.95	1.0	Tampa
-77.03	38.88	1.0	Washington

sequence: 1) geographical longitude, 2) geographical latitude, 3) relative weight, and 3) site identification string (e.g., its name). The geographical angles must be given in degrees. An example is given in Table XIV.

If the selected sites input file is `Random_sites`, then the sites are randomly generated. This specific input file is made of three lines. In the first line the number of random sites to generate and the distribution type of the associated weights are given. Two distribution type are available. The first one assigns the same identical weight to all the sites. Thus, the relative weight of each site will be $1/N$, where N is the number of the sites to be generated. A second distribution type provides weight with uniform probability distribution. The second and third lines contain the longitude and latitude ranges, respectively. For example, in order to create four random sites with relative weights uniformly distributed and located within the longitude range $[12.0°, 333°]$ and latitude range $[-27°, 45.7°]$, has the structure shown in Fig. 62.

2. *Orbit input file.* Any file with the proper format, of extension `.ind` is accept-

```
  4    -5      No. sites & weight distr. (0=const; random)
 12.0 333      Longitude range in degrees [0, 360]
-27    45.7    Latitude  range in degrees [-90, 90]
```

Fig. 62. Structure of the input file defining the target sites.

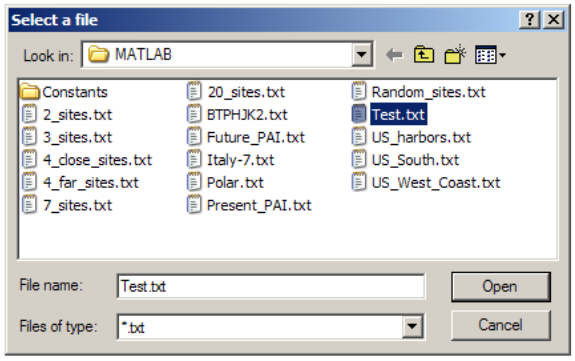
able. This file contains two-lines of text only. The first is the time, provided in calendar form (year, month, day, hour, minute, and seconds respectively). The second line contains the orbital elements in the following order: 1) semi-major axis, 2) eccentricity, 3) orbit inclination, 4) argument of perigee, 5) right ascension of the ascending node, and 6) true anomaly. All the angles must be in radiant and the distance in kilometers. A typical *Orbit input file* is shown in Fig. 63.

```
2007   3  31  23   59  59.0   % Initial time [Y,M,D,h,m,s]
14416.8344 0.1 1.57 0.0 0.0 0.0  % Orb.El. [a,e,i,w,RA,TA]
```

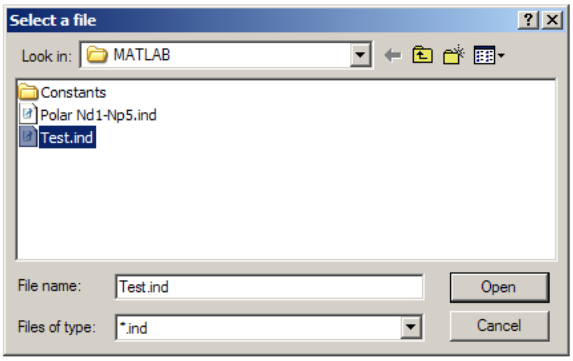
Fig. 63. Structure of the orbit input file for program J2Exploitation.

The `J2Exploitation` program prompts the sequence of dialog windows shown in Figs. 64 and 65 allowing the user to select/modify default parameters. The program provides also in output the sequence of plots shown in Figs. 66 and 67 which show the ground track and the observations before and after the maneuver at time T_m, respectively.

The program also provides reporting data to the console; an example is shown in Fig. 68.

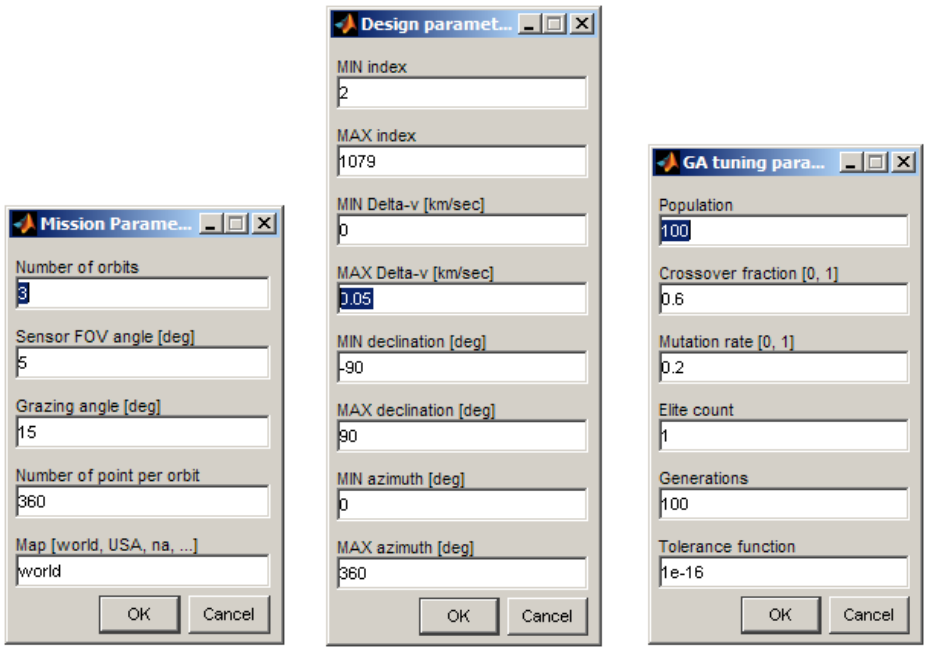


(a) Input file selection window.



(b) Orbit file selection window.

Fig. 64. J2Exploitation program file input windows.



(a) Input of mission parameters. (b) Gene ranges window. (c) GA Tuning Parameters.

Fig. 65. Parameter input dialogs for the J2Exploitation program.

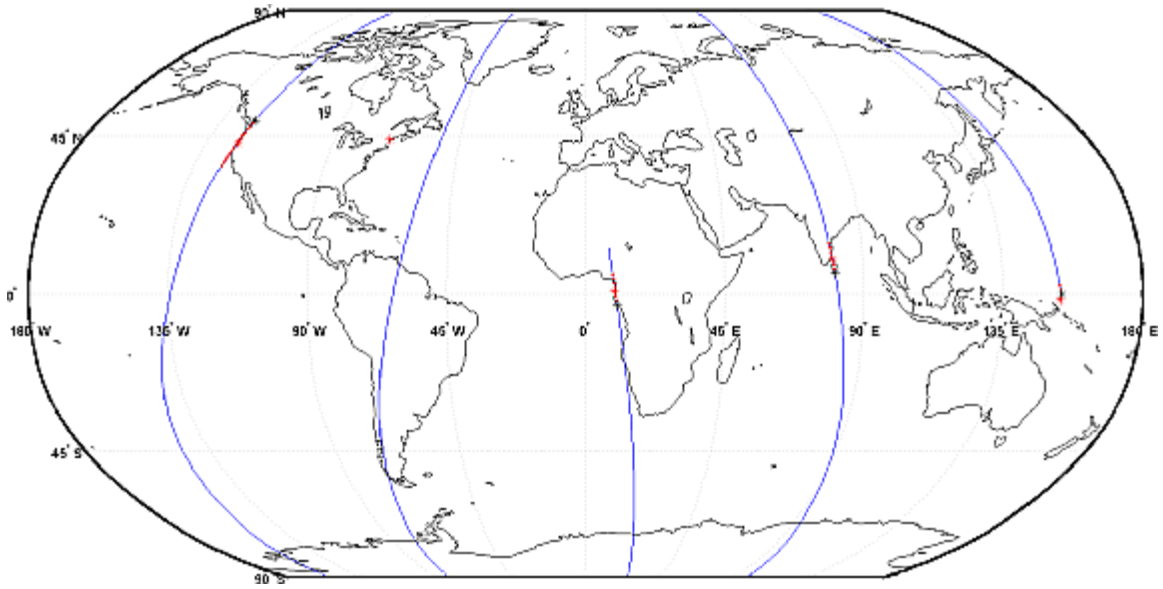


Fig. 66. J2Exploitation program output: Ground track and observations before maneuver.

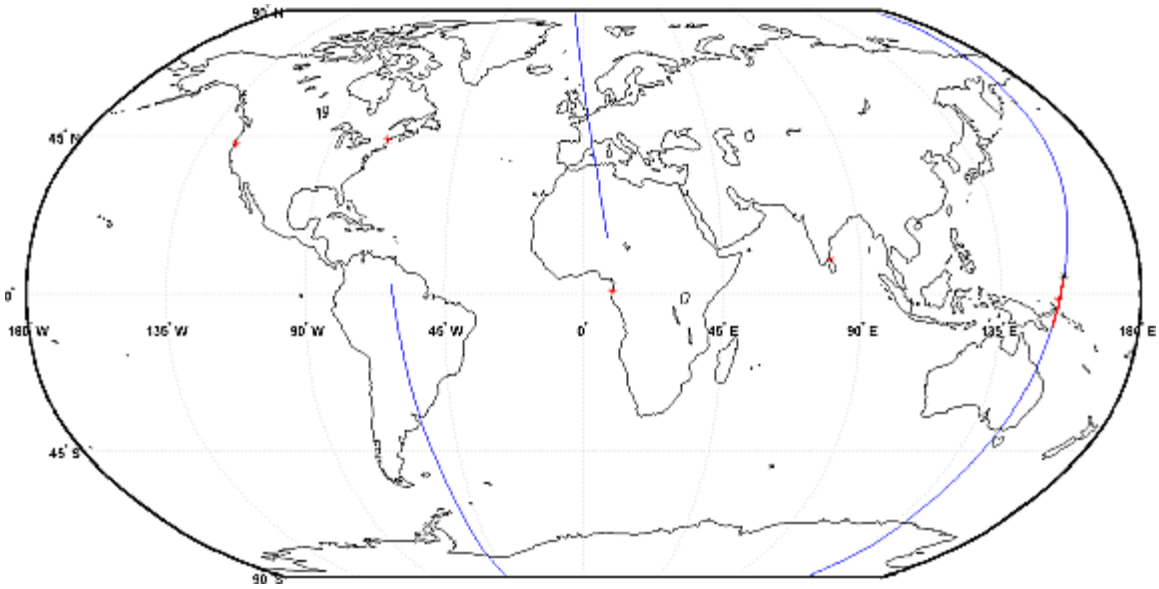


Fig. 67. J2Exploitation program output: Ground track and observations after maneuver.

```
====>   Mission parameters  <=====
Number of orbits = 3

====>   Initial Orbital Parameters  <=====
I_Time = 03-Mar-2007 03:03:03   (JD=2454162.6271)
F_Time = 03-Mar-2007 17:24:24   (JD=2454163.2253)
     T = 4.7853 hr
   sma = 14416.8344 km
   ecc = 0.1
   inc = 90 deg
 omega = 0 deg
  RAAN = 0 deg
TrueAn = 0 deg
Optimization terminated: average change
in the fitness value less than options.TolFun.
   Cost function = -596.3153
   Impulse index = 734

====>   Impulse Parameters  <=====
Delta-v = 2.3579e-005 km/sec
Delta-v time = 03-Mar-2007 12:45:46   (JD=2454163.0318)

====>   Before/After the impulse  <=====
  Orbital period :   4.78535   4.78542 hr
 Semi-major axis : 14416.834 14416.975 km
    Eccentricity : 0.1000000 0.1000088
     Inclination :   90.0000   90.0000 deg
Argument perigee :  359.8815  359.8822 deg
        R.A.A.N. :    0.0000  360.0000 deg
    True Anomaly :   13.0362   13.0356 deg
```

Fig. 68. Program J2Exploitation output.

CHAPTER VII

CONCLUSIONS

The main contribution of this dissertation are 1) the development of reusable software that will be enhanced and expanded by those who will continue this line of research and 2) the presentation of a set of practical applications in which the Flower Constellations can be utilized instead, or more likely together, with the classic constellation design methods.

The past work on Flower Constellations concentrated on establishing the rigorous mathematical foundations of this design methodology and like solving a puzzle every new piece in place opened up new possibilities and new challenges.

It had come a time, however, in which the question "What can we do with this new tool?" became pressing, not only in the minds of those actively involved in the research, but the same question was being also asked from those interested in learning about this new constellation concept.

The first, painful but necessary step has been that of developing the software, the basic building blocks that would allow us to examine and explore the possibilities. The first part of the software development, the FCVAT program, was actually done during the first phase of the Flower Constellation development, some three years ago, and is still going on now, as new ideas and enhancements come forward. The need of expanding the software currently availably for fast prototyping orbits, satellites, and Earth Observation systems in MATLAB was felt for a long time, until the development of the FCToolbox was started with the decision of making it part of this dissertation.

Once the basic building blocks have been built the attention was concentrated on applying the Flower Constellations to missions of interest for several communities:

Earth Observation, Telecommunications, Global Navigation, and Reconnaissance.

This work is just another step, most likely, and hopefully, not the last to involve Flower Constellations. The software is growing but is far from satisfying all the needs: one would like to have the possibility of plotting ground tracks from within the FCVAT program itself, and the possibility to extend the tool to become a general satellite analysis tool. The FCToolbox for Matlab could be extended to become a full fledged Celestial Mechanics Toolbox, which is sourly missing from the MATLAB software offer.

The applications for Flower Constellations are far from being exhausted. If accessibility to real mission needs is provided there is hope that we could see a Flower Constellation operational concept in the next decade.

In conclusion it is the voyage that matters, and the Flower Constellation voyage is not really finished yet; it is only just started.

REFERENCES

[1] B. Parkinson and J. Spilker Jr., *Global Positioning System: Theory and Applications*, vol. I-II, Washington, DC: American Institute of Aeronautics and Astronautics (AIAA), 1996.

[2] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*, 4th ed., New York: Springer-Verlag, 1997.

[3] J. Farrell and M. Barth, *The Global Positioning System and Inertial Navigation*, New York: McGraw-Hill, 1998.

[4] United States Coast Guard. (2007), "Gps almanac information," [ONLINE]. AVAILABLE: http://www.navcen.uscg.gov/gps/almanacs.htm.

[5] European Commission. (2007), "GALILEO - european satellite navigation system," [ONLINE]. AVAILABLE: http://ec.europa.eu/dgs/energy_transport/galileo/index_en.htm.

[6] J. G. Walker, "Some circular orbit patterns providing continuous whole earth coverage," *Journal of the British Interplanetary Society (JBIS)*, vol. 24, pp. 369–384, 1971.

[7] J. G. Walker, "Satellite constellations," *Journal of the British Interplanetary Society (JBIS)*, vol. 37, pp. 559–572, 1984.

[8] J. R. Wertz, H. F. Messinger, L. K. Kraft Newman, and G. N. Smit, *Mission Geometry; Orbit and Constellation Design and Management*, El Segundo CA: Microcosm Press and Boston MA: Kluwer Academic Press, 2001.

[9] J. E. Draim, "Elliptical orbit MEO constellations: A cost-effective approach for multi-satellite systems," *Space Technology*, vol. 16, no. 1, pp. 21–29, 1996.

[10] R. Proulx, J. Smith, J. E. Draim, and P. Cefola, "Ellipso gear array - coordinated elliptical/circular constellations," in *Procedeengs of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Boston, MA, Aug. 10-12, 1998*, 1998.

[11] J. E. Draim, "Three - and Four-Satellite Continuous-Coverage Constellations," *Journal of Guidance, Control and Dynamics*, vol. 8, no. 6, pp. 725–730, 1985.

[12] J. E. Draim, R. Inciardi, P. Cefola, R. Proulx, and D. Carter, "Demonstration of the COBRA teardrop concept using two smallsats in 8-hr elliptic orbits," in *Procedeengs of Fifteenth AIAA/USU Conference on Small Satellites, Logan, UT*, 2001.

[13] J. E. Draim, "Satellite constellations," in *The Breakwell Memorial Lecture, 55th International Astronautical Congress, IAC-04-A.5.01, Vancouver, Canada*, 2004.

[14] W. J. Mason, V. Coverstone-Carroll, and J. W. Hartmann, "Optimal earth orbiting satellite constellation via a Pareto genetic algorithm," in *Proceedings of AIAA/AAS Astrodyn. Spec. Conf., paper no. 98-4381, Boston, Mass.*, 1998.

[15] J. E. Smith, "Application of optimization techniques to the design and maintenaince of satellite constellations," M.S. thesis, Cambridge, MA, Massachusets Institute of Technology, 1999.

[16] M. Asvial, R. Tafazolli, and B. G. Evans, "Satellite constellation design and radio resource management using genetic algorithm," *IEEE Proc.-Commun.*,

vol. 151, no. 3, pp. 204–209, 2004.

[17] C. Bruccoleri and D. Mortari, "The Flower Constellations visualization and analysis tool," in *2005 IEEE Aerospace Conf.*, Big Sky, MT, March 5-12, 2005.

[18] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, New York: AIAA Education Series, revised edition, 1999.

[19] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*, vol. 2, New York: McGraw-Hill, 2001.

[20] H. P. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*, Reston, VA: American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2003.

[21] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, pp. 141–158, New York: AIAA Education Series, revised edition, 1999.

[22] P. Colwell, *Solving Kepler's Equation over Three Centuries*, Richmond, VA: Willmann-Bell, 1993.

[23] A. Nijenhuis, "Solving Kepler's equation with high efficiency and accuracy," *Celestial Mechanics and Dynamical Astronomy*, vol. 51, pp. 319–330, 1991.

[24] F. L. Markley, "Kepler equation solver," *Celestial Mechanics and Dynamical Astronomy*, vol. 63, pp. 101–111, 1996.

[25] T. Fukushima, "A method solving kepler's equation without transcendental function evaluations," *Celestial Mechanics and Dynamical Astronomy*, vol. 66, pp. 309–319, 1997.

[26] S. A. Feinstein and C. A. McLaughlin, "Dynamic discretization method for solving kepler's equation," *Celestial Mechanics and Dynamical Astronomy*, vol. 96, pp. 49–62, September 2006.

[27] D. Mortari and A. Clocchiatti, "Solving Kepler's equation using Beziér curves," in *Proceedings of the 7th Dynamics and Control of Systems and Structures in Space Conference*, July London, UK, July 16-20, 2006.

[28] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, pp. 471–490, New York: AIAA Education Series, revised edition, 1999.

[29] D. Mortari, M. P. Wilkins, and Bruccoleri C., "The Flower Constellations," in *Proceedings of The John L. Junkins Astrodynamics Symposium*, Paper AAS 03-274, Texas A&M University, College Station, TX, 2003.

[30] M. Avendano and D. Mortari, "Rotating symmetries in space: The flower constellations," To be published, February 2007.

[31] J. R. Wertz, H. F. Messinger, L. K. Kraft Newman, and G. N. Smit, *Mission Geometry; Orbit and Constellation Design and Management*, pp. 683–685, El Segundo CA: Microcosm Press and Boston MA: Kluwer Academic Press, 2001.

[32] Wikipedia - The free encyclopedia. (2007), "Iridium satellite constellation," [ONLINE]. AVAILABLE: http://en.wikipedia.org/wiki/Iridium_satellite.

[33] GLOBALSTAR, Inc. (2007), "About globalstar," [ONLINE]. AVAILABLE: http://www.globalstarusa.com/en/content.php?cid=601.

[34] P. A. Monte and A. E. Turner, "Constellation selection for globalstar, a global mobile communications system," in *Proccedings of the AIAA International Communication Satellite Systems Conference and Exhibit*, 1992, pp. 1350–1360.

[35] D. Mortari and M. P. Wilkins, "The flower constellation set theory part I: Compatibility and phasing," *IEEE Transactions on Aerospace and Electronic Systems*, To be published 2008.

[36] M. P. Wilkins, "The Flower Constellations - theory, design process, and applications," Ph.D. dissertation, Dep. of Aerospace Eng., Texas A&M University, College Station, TX, 2004.

[37] M. P. Wilkins and D. Mortari, "The Flower Constellation set theory part II: Secondary path and equivalency," *IEEE Transactions on Aerospace and Electronic Systems*, To be published 2008.

[38] O. O. Abdelkhalik and Mortari D., "Two-way orbits," *Celestial Mechanics and Dynamic Astronomy*, vol. 94, no. 4, pp. 399–410, April 2006.

[39] D. Mortari and M. P. Wilkins, "Dual-compatible Flower Constellations," in *Proceedings of the 2006 Space Flight Mechanics Meeting Conference*, Paper AAS 06-202, 2006.

[40] C. Wagner, "Prograde geosat exact repeat mission?," *The Journal of the Astronautical Sciences*, vol. 39, pp. 313–326, July-September 1991.

[41] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*, Boca Raton, FL: Chapman & Hall/CRC, 2004.

[42] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*, Washington DC: Hemisphere/Wiley, 1975.

[43] Wikipedia - The free encyclopedia. (2007), "Evolutionary Algorithms," [ONLINE]. AVAILABLE: http://en.wikipedia.org/wiki/Evolutionary_algorithm.

[44] D.H. Wolpert and W.G. Macready, "No free lunch theorems for search," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[45] L. Davis, *Genetic Algorithms and Simulated Annealing*, Los Altos, CA: Morgan Kaufmann, 1987.

[46] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE Int. Conf. on Neural Networks*, Piscataway, NJ, 1995, p. 19421948.

[47] The Mathworks, Inc. (2007), "Documentation of genetic algorithm and direct search toolbox," [ONLINE]. AVAILABLE: http://www.mathworks.com/access/helpdesk/help/toolbox/gads/gads.shtml.

[48] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effect of noise," Tech. Rep., Department of General Engineering, University of Illinois, Urbana-Champaign, IL, 1995.

[49] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, New York: John Wiley & Sons, 1966.

[50] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, pp. 27–57, Reading, MA: Addison-Wesley, 1989.

[51] G. Syswerda, "Uniform crossover in genetic algorithms," in *J. D. Schaffer Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp. 2–9.

[52] D. B. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence.*, New York: IEEE Press, 2000.

[53] Sun Microsystems, Inc. (2007), "Sun Developer Network," [ONLINE]. AVAILABLE: http://java.sun.com.

[54] Eclipse Consortium. (2007), "Eclipse - an open development platform," [ONLINE]. AVAILABLE: http://www.eclipse.org.

[55] OpenGL.org. (2007), "OpenGL home page," [ONLINE]. AVAILABLE: http://www.opengl.org.

[56] Object Management Group - OMG. (2007), "Object Management Group, Unified Modeling Language, UML," [ONLINE]. AVAILABLE: http://www.uml.org/.

[57] Java3D Organization. (2007), "Java3D Home Page," [ONLINE]. AVAILABLE: http://java3d.j3d.org/, Hosted by Yumetech, Inc.

[58] MPlayer Development Team. (2007), "MPlayer - The Movie Player," [ONLINE]. AVAILABLE: http://www.mplayerhq.hu/design7/info.html.

[59] Wikipedia - The free encyclopedia. (2007), "Geoid," [ONLINE]. AVAILABLE: http://en.wikipedia.org/wiki/Geoid.

[60] J. D. Turner, "A non-iterative & non-singular perturbation solution for transforming cartesian to geodetic coordinates," To be published, May 2007.

[61] The Mathworks, Inc. (2007), "On line help for the Mapping Toolbox," [ONLINE]. AVAILABLE: http://www.mathworks.com/access/helpdesk/help/toolbox/map/index.html.

[62] J. R. Wertz, H. F. Messinger, L. K. Kraft Newman, and G. N. Smit, *Mission Geometry; Orbit and Constellation Design and Management*, pp. 418–426, El Segundo CA: Microcosm Press and Boston MA: Kluwer Academic Press, 2001.

[63] University Corporation for Atmospheric Research (UCAR). (2007), "The earth's magnetic field," [ONLINE]. AVAILABLE: http://www.windows.ucar.edu/tour/link=/earth/images/earth_magneto_image.html&edu=high.

[64] European Space Agency. (2007), "Diagram showing the van allen radiation belts," [ONLINE]. AVAILABLE: http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=38475.

[65] K. J. Park, M. P. Wilkins, and D. Mortari, "Uniformly distributed Flower Constellation design study for Global Positioning System," in *Proceedings of the 2004 Space Flight Mechanics Meeting Conference*, Maui, Hawaii, Paper AAS 04-297, 2004.

[66] R. C. Olsen, *Remote Sensing from Air and Spaces*, Bellingham WA: SPIE press, 2007.

[67] National Reconnaissance Office (NRO). (2007), "Corona spacecraft camera technology," [ONLINE]. AVAILABLE: http://www.nro.gov/corona/imagery.html.

[68] National Reconnaissance Office (NRO). (2007), "Corona spacecraft camera technology," [ONLINE]. AVAILABLE: http://www.nro.gov/corona/imagery.html.

[69] National Reconnaissance Office (NRO). (2007), "Corona spacecraft camera technology," [ONLINE]. AVAILABLE: http://www.nro.gov/corona/imagery.html.

[70] JPL/NASA. (2007), "SIR-C/X-SAR image of mount Etna," [ONLINE]. AVAILABLE: http://www.jpl.nasa.gov/radar/sircxsar/etna.html.

[71] O. O. Abdelkhalik, "Orbit design and estimation for surveillance using Genetic Algorithms," Ph.D. dissertation, Dep. of Aerospace Eng., Texas A&M University, College Station, TX, 2005.

APPENDIX A

CONTENTS OF THE FCTOOLBOX

A list of the functions contained in each package of the FCToolbox. The toolbox is initialized with the script `Init` that adds the path of each package to the MATLAB search path. This kind of initialization is done to decouple, as much as possible, the FCToolbox from the rest of the environment. If wanted, one can set MATLAB to run the script automatically at startup, thus maximum flexibility is provided. The notation A::B stands for "package B is included in package A", the '::' (double column) symbol is called scope operator (in UML and C++). The list of functions below is simply a reference of what is included in the FCToolbox, a much more detailed reference documentation is available in HTML hypertext format. Classes have slightly different naming conventions: functions acting on class objects are called *methods* and the fields of classes and structures are called *attributes* in the OOP vernacular.

FCTOOLBOX Flower Constellation Matlab Toolbox

`Init` This script initializes the Flower Constellation Toolbox path.

`get_fctoolbox_path` Return the path where the FCToolbox is installed.

FCTOOLBOX::COORDINATES

`ecef2sez` Convert ECEF coordinates to topocentric (SEZ) frame.

`ecef2sez_dcm` Compute transformation from ECEF to SEZ coordinates.

`ecf2eci` ECF to ECI coordinates transformation.

`eci2ecef` ECI to ECF coordinate transformation.

`eci2ecf` ECI to ECF coordinate transformation.

`eci2lvlh` Convert eci unit pointing vector to local vertical local horizon.

`eci2topo` Computes topocentric (SEZ) coordinates and rates from satellite ECI position and velocity.

`enu2ned` Convert coordinates from ENU to NED frames.

`enu2ned_dcm` Compute DCM to convert coordinates from ENU to NED frames.

`enu2sez` Convert coordinates from ENU to SEZ frames.

`enu2sez_dcm` Compute DCM to convert coordinates from ENU to SEZ frames.

`gsite` Compute earth site position vector in geocentric (ECEF) coordinates.

`ned2sez` Convert coordinates from NED to SEZ frames.

`orb2eci` Converts classical orbital elements to ECI state vector.

`sez2eci` Converts topocentric coordinates (SEZ) to ECI position and velocity vectors.

`sez2elazrng` Compute elevation, azimuth and slant.

`sez2ned` Convert coordinates from SEZ to NED frames.

`topo2eci` Converts topocentric coordinates to ECI position and velocity vectors.

`wgs84_ellipsoid` Return the reference ellipsoid wgs84 as a row vector.

FCTOOLBOX::NAVIGATION

`decode_4vars` Decode chromosome X from GA and return a FC object.

`decode_incl` Decode chromosome X from GA and return a FC object.

`dop` Compute Dilution of Precision (DOP) for a GPS receiver.

`dop_by_latitude` Compute GDOP at different latitudes.

`dop_compare` Compute GDOP for a given constellation at different latitudes.

`dop_scan` Compute DOP of a constellation for receivers at specified latitudes.

`fcsymoptions` Generate default options for sym_fc programs.

`globcov_fit_gdop` Fitness function for global coverage FC.

`globcov_fitness_fcn` Fitness function for global coverage FC.

`J2_prop_eci` Analytic orbit propagation, J2 linear propagation ECI version.

`J2_prop_init` Initializes global variables needed by the J2_prop_ecef function.

`los_plot` Simple 3D LOS plot of a constellation.

`plot_energy` Plot potential energy for a Flower Constellation

`print_iter` print_iter custom OutputFcn for GA.

`sat_dop` Compute Dilution Of Precision (DOP) quantities for a set of satellites.

`show_max_gdop` Show the configuration with worst GDOP

`sym_fc1` FC Optimization for Global Navigation, prog 1.

`sym_fc2` FC Optimization for Global Navigation, prog 2.

`sym_fc3` FC Optimization for Global Navigation, prog 3.

`sym_fc4` FC Optimization for Global Navigation, prog 4.

`sym_fc5` FC Optimization for Global Navigation, prog 5.

FCTOOLBOX::NUMERICAL

`compatible_sma` Compute the semi-major axis a for a compatible orbit

`compatible_sma_hp` Compute the semi-major axis a for a compatible orbit

`compute_nso` Compute the maximum number of satellites per orbit in FC

`constrained_config` Return a FC matrix of valid configurations that satisfy the constraint string.

`coprime` Return all numbers Q in q that are coprime with p

`divisors` All possible divisors.

`FindNpNd` Generate co-prime pairs of Np and Nd integers parameters in FCs.

FCTOOLBOX::RECONNAISSANCE

`atan3` Computes the angle when sin and cos are known.

`Compatible_sma` Computes the semi-major axis of compatible orbits.

`coverage` Compute coverage of multiple satellites over multiple sites during a specified interval of time.

`cw` Constant: ratio degrees/radians.

`EarthRadius` Constant: Earth equatorial radius, [km], WGS84

`EarthSpinRate` Constant: Earth sidereal rotation rate [rad/sec]

`eci2orb` Converts ECI radius and velocity to orbital elements

`fc-seq` Generates satellites sequences of a FC.

`FitFun1` Fitness function for Main1.m

`FitFun2` Fitness function for Main2.m

`FitFun3` Fitness function for Main3.m program.

`gapcount` Counts the number of discontinuous clusters of "1" in a logical vector v.

`gast` Compute Greenwich (apparent) Sidereal Time (gst)

`gdate` Converts Julian date to Gregorian (calendar) date.

`ground_track_anim` Constellation ground track and coverage analysis.

`J2` Constant: J2 linear term.

`J2_prop_ecef` Analytic orbit propagation, J2 linear propagation ECEF version.

`J2_prop_eci` Analytic orbit propagation, J2 linear propagation ECI version.

`J2_prop_init` Initializes global variables needed by the J2_prop_ecef function.

`jd2str` Converts Julian date to string equivalent calendar date (UTC).

`julian_old` Evaluates the Julian date of a calendar date (UTC).

`kepler_danby` Solves Kepler's equation using Danby's method.

`kepler_danby_mikkola` Solve Kepler's equation using Danby's method with Mikkola's initial guess.

`LatLon2Az` Evaluates the azimuth of site #2 with respect to site #1.

`Main1` Reconnaissance Main Program #1.

`Main2` Reconnaissance Main Program #2.

`Main3` Reconnaissance Main Program #3.

`mu` Earth Gravitational Constant.

`orb2eci` Converts classical orbital elements to ECI state vector

`Per_Maint_Cost` Perigee Maintenance costs using radial impulse per year

`pidiv2` Constant: pi/2

`Plot_Obs1` Plots trajectories and observation characteristics.

`Plot_Orbits_and_Sites` Plot the results.

`Plot_Results1` Plot the results for Main1.m.

`r2r` Revolutions to radians function.

`Read_Orbit` Reads orbit data file for "Main2.m" Program

`ReadSites` Reads Earth site data files.

`rec2pol` Converts Rectangular to Polar coordinates.

`resolution` Compute maximum theoretical ground resolution of an imaging system given by Rayleigh criterion.

`Rx` Builds the rotation matrix about the "x" axis

`Rz` Builds the rotation matrix about the "z" axis

`ShowObs1` This function plots the main characteristics of the observation passage.

`Simulate_Optimal` Simulation program for Reconnaissance of Earth sites from space.

`sumcoverage` Computes coverage characteristics.

`sun` Solar ephemeris.

`swat_ang` Compute the Earth Central Angle and swat given the range and the sensor half FoV angle.

`swat_lines` Compute the Earth Central Angle and swat given the range and the sensor half FoV angle.

`swat_patch` Draw the swat defined by the input arrays on the current map axes.

`ta2ea` Converts True Anomaly in Eccentric Anomaly

`ta2ma` Converts true anomaly in mean anomaly

`twopi` Constant: 2*pi

`visible_spectrum` Returns wavelength limits of the visible spectrum in nm.

FCTOOLBOX::Test

`angle_test` Generate plots of the angle between the Sub Satellite Point.

`dop_test` Tests for DOP routines.

`fc_gamma` This function computes the Flower Constellation angle "gamma".

`movies` Generate movies of FC coverage.

`orb_plane_test` Draw several transparent orbital planes. Test for figure generation.

`test_arrow` plot an arrow head

`test_coverage` Test the coverage routines with the Broglio FC

`test_fc_uniform` Test generation of uniform FC.

`test_globe` Test plotting of Earth globe.

`test_ground_track` Constellation ground track and coverage analysis.

`test_orbit_plots` Test the use of functions for plotting orbits.

`test_propag` Test for orbital propagation.

`test_recon` Reconnaissance problem test program.

`test_retrograde` Test retrograde uniformly distributed FC.

`test_walker` Test generation of Walker constellations.

FCTOOLBOX::Utility

`arrow3d` Plot a 3D arrow head using patch objects.

`arrow_head3d` Plot a 3D arrow head using patch objects.

`dop_analysis` Perform Dilution of Precision (DOP) Analysis on a set of satellites.

`earth_globe` Draw the Earth Globe on mapping axes.

`ecef_position` Calculate ECEF position of a satellite.

`eci2orb` Convert ECI state vector to six classical orbital elements.

`ellipse` Compute the coordinates of an ellipse centered at its focus.

`equatorial_plane` Draw the equatorial plane on the current axes.

`fc_equiv` This function checks whether two Flower Constellations are equivalent.

`fc_seq` Build the RAAN and M sequences for the specified FC.

`geodet_points` Return longitude and latitude of n points evenly distributed on a Earth like spheroid.

`get_admissible_integers` Find all possible FC configurations of the integers parameters.

`is_uniform` Return true if the satellites in the FC are uniformly distributed.

`line2pts` Draw a 3D line between two points on the current axes.

`nominal_galileo` Returns the classical orbital elements for the nominal GalileoSat constellation.

`nominal_GLONASS` Returns the nominal orbital elements of the GPS constellation.

`nominal_GPS` Returns the nominal orbital elements of the GPS constellation.

`orbit_plane` Draw the orbital plane ellipse with the given value of color and transparency.

`orbit_plot` Plot the orbit ellipse given the Orbital Elements.

`period` Computes the period of an earth orbiting satellite.

`plot_dop` Plot DOP analysis graphs.

`plot_ground_track` Plot ground track on the desired map projection.

`points` Uniform distribution of points on a d dimensional sphere.

`potential_energy` Compute potential energy of a configuration of points.

`quick_earth` Cartesian coordinates of a sphere with radius equal to Earth equatorial radius.

`ref_frame` Draws reference frame axes on the current figure.

`remove_collisions` Remove Flower Constellations that have collisions between satellites.

`remove_non_uniform` Remove non uniform constellation from the Configurations Matrix FC.

`walker` Build a Walker constellation for the specified parameters.

`WalkerDelta` Compute the Orbital Elements of satellites belonging to a Walker Delta Pattern.

`gdtrackanim_options` Build an options structure to be used by ground_track_anim().

Class FCTOOLBOX::FlowerConstellation

`char` Convert FC parameters to a string.

`compute_nso` Compute the maximum number of satellites per orbit in a FC.

`disp` Display FC parameters on console.

`display` Overload: FlowerConstellation/display

`dop_analysis` FlowerConstellation/dop_analysis Perform Dilution of Precision (DOP) Analysis on FC satellites.

`FlowerConstellation` Constructor: generates orbital elements of the satellites in the FC.

`get_orb_elem` This function returns a vector of OrbitalElements objects for the satellites in a FC.

`is_uniform` FlowerConstellation/is_uniform Return true if the satellites in the FC are uniformly distributed on the relative trajectory.

`period` FlowerConstellation/period - Computes the orbital period for a FC.

`print` FlowerConstellation/print Print information data on a FC.

`quick_ground_track` Plot ground track of the FC on a mercator map.

`quick_view` Simple 3D plot of the flower constellation.

`save` Save a .flower text file compatible with FCVAT.

`subsasgn` OVERLOAD: FlowerConstellation/subsasgn

`subsref` OVERLOAD: FlowerConstellation/subsref

`test` FlowerConstellation/test Unit tests for the FC class.

Class FCTOOLBOX::OrbitalElements

`char` String conversion operator.

`deg2rad` Converts angles in OE to radians.

`disp` OVERLOAD: disp

`display` OVERLOAD: OrbitalElements/display

`double` Convert an array of OrbitalElements to an array of double.

`orb2eci` Converts classical orbital elements to ECI state vector.

`OrbitalElements` Class Constructor.

`rad2deg` Converts angles in OE to degrees.

`subsasgn` OVERLOAD: subasgn.

`subsref` OVERLOAD: subsref.

`test` Unit tests for class OrbitalElements

Class FCTOOLBOX::Planet

`char` Converts a planet to a string

`disp` planet/disp

`display` planet/display

`double` planet/double

`Planet` Constructor for class planet.

`subsasgn` OVERLOAD: subasgn.

`subsref` OVERLOAD: subsref.

VITA

Christian Bruccoleri received his Laurea degree (M.S.) in Computer Science from University of Rome, La Sapienza, in 2002. He entered the Aerospace Engineering program at Texas A&M University in the same year. His research interests include satellite constellations, computer vision, autonomous star pattern recognition and attitude determination. He plans to join StarVision Technologies, Inc. from January 2008, where he will continue to work on the same research topics. He can be reached at StarVision Technologies Inc., 1700 Research Parkway, Suite 170, College Station, TX 77845. His email is cbruccoleri@gmail.com.

The typist for this dissertation was Christian Bruccoleri.