Analysis of Relative Navigation Architectures for Formation Flying Spacecrafts

by

Henry Jacques Lefebvre de Plinval-Salgues

Diplôme d'Ingénieur de l'Ecole Polytechnique, 2004

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2006

Author ....
Department of Aeronautics and Astronautics
January 31, 2006

Certified by .....
Jonathan P. How
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by ....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Analysis of Relative Navigation Architectures for Formation Flying Spacecrafts

by

Henry Jacques Lefebvre de Plinval-Salgues

Submitted to the Department of Aeronautics and Astronautics
on January 31, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Many future space missions will involve fleets with a large number of satellites flying in formation. Indeed, such fleets provably offer more reliability, redundancy, scalability and repeatability. However, large fleets also represent a challenge, especially for the navigation algorithms, which must provide an accurate estimate of the state of the fleet, with minimum requirements. Furthermore, as the number of satellites in the fleet increases, the computations to be performed increase dramatically, as well as the synchronization and communication requirements, making the design of efficient algorithms a difficult challenge. Based on previous studies, Decentralized Algorithms were designed to spread the computational task. Hierarchic Algorithms were also studied in order to reduce the synchronization requirements. This thesis presents both analytical and numerical comparisons of these algorithms in terms of accuracy, computational complexity, synchronization, and communication. The Decentralized and Hierarchic Algorithms were shown to have good performance in terms of accuracy, while involving far fewer computations than the Centralized Algorithm. As a result, they can be used as scalable algorithms for large formation flying fleets. The thesis investigated two additional problems often associated with navigation filters. The first study considers the problem of processing delayed measurements. Three strategies are analyzed, and compared in terms of the accuracy of the estimate they perform, and the memory and computations they require. One of these approach is shown to be efficient, being accurate without requiring heavy computations nor memory. The second study analyzes a particular instability of the Extended Kalman Filter, encountered when two sensors have very different accuracies. The instability is explained and a method to fix it is proposed. In the example analyzed the method proves to be efficient in addressing the instability.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

In the course of this research, several people were of great support. I want to mention first my beloved fiancee Floraine whose love is my constant motivation for improvement. My parents and my brothers helped me become the adult I am, and I thank them warmly for that. I would like to thank Professor Jonathan How for his constant drive for rigor during this research, and his many helpful insights. I want to thank my labmates for their help, Dr. Gokhan Inalhan, Louis Breger, Milan Mandic, Yoshiaki Kuwata, Luca Bertuccelli, Thomas Chabot, Mehdi Alighanbari, Ian Garcia, Justin Teo, Han-Lim Choi, Byunghoon Kim. I would like to thank as well the French community–I cannot say all the names here–for the charming furia francese they bring to Boston. I would like also to thank all the great people I have met here: I cannot write all the names, but they are in my mind.

This research was funded under NASA Space Communications Project Grant #NAG3-2839. I thank Professor How and NASA for their continued funding through this research project.

GAUDET QUI POTUIT RERUM COGNOSCERE CAUSAS

...

SED BEATUR QUI POTEST HUMANORUM COGNOSCERE CAUSAM !

Henry de Plinval

Contents

List of Figures

Chapter 1

Introduction

A key feature of space missions is scalability, as well as reconfigurability and robustness to failures. For this reason, a general trend of space missions is to rely on a constellation of small satellites rather than one unique monolithic spacecraft, whose failure may have catastrophical consequences [1, 2, 3]. This change does not come without challenges, especially in the area of navigation algorithms. Indeed, estimating the relative and/or absolute state of satellites in a large fleet requires an increasing amount of computation, communication and synchronization to maintain the performance (accuracy) as the number of satellites increases.

In this context, various algorithms have been proposed and studied [4, 5, 6, 7, 8]. However, it was often found that improving an algorithm with respect to one of these criteria would typically decrease its performances in another area, so that the various algorithms are all Pareto-optimal solutions. Designing algorithms better than the existing ones with respect to all criteria is still a challenge.

1.1 Research Objectives

The main goal of this research was to assess the comparative qualities of various algorithms, Centralized, Decentralized and Hierarchic, in terms of their accuracy,

computational complexity, synchronization requirements, and communication load. This comparison was performed first through analytical studies. Then, the algorithms were implemented, and numerical comparisons were held. Former studies [4, 5] had already assessed some of the Decentralized Algorithms, and proposed the Hierarchic Scheme to further improve the scalability of the estimate. The research presented here follows these studies in that it compares these algorithms, included the Hierarchic Algorithms, both analytically and numerically.

Beyond this main topic, two particular problems encountered in navigation algorithms for formation flying satellites were also investigated. First, the problem of integrating delayed measurements in an estimation algorithm was addressed, in the context of the Magnetospheric Multiscale Mission [6, 9]. Three strategies to use these delayed measurements were considered and evaluated.

Second, a particular instability of the Extended Kalman Filter, raised when two sensors have very different accuracies, was studied. The reason for this instability was described, intuitively, by walking through a simple example, and with analytical tools. A solution to improve the stability of the Extended Kalman Filter in this context was proposed and evaluated.

1.2 Thesis Outline

In Chapter 2, observability studies are performed for various measurements, frames and states scenarios. One observable set-up is chosen for the subsequent studies.

Chapter 3 presents the Centralized Algorithm to solve the estimation issue. It emphasizes the limitations of this algorithm, and introduces the Decentralized Scheme to address these issues. The Centralized Algorithm is qualitatively compared with various Decentralized Algorithms.

Chapter 4 presents Hierarchic Architectures as a way to further improve the estimation task. Various algorithms–Centralized, Decentralized and Hierarchic–are com-

pared, both analytically by use of simplified models, and numerically.

In Chapter 5, a particular issue raised in the Magnetospheric Multiscale Mission is studied. Three approaches to integrate delayed measurements in the estimation process are investigated and compared.

Chapter 6 analyzes another particular issue. When two sensors of very different accuracies are used, the Extended Kalman Filter shows great instability. An explanation for this instability is provide, and a method to fix it is proposed.

Chapter 2

Preliminary Studies for Measurement Infrastructure Determination

2.1 Goal and Challenges

A key component of a navigation architecture is the sensor infrastructure used to provide measurements on the fleet state. When designing efficient navigation algorithms, one wishes to use as accurate and thorough measurements as possible, in order to improve the estimation accuracy. However, because of cost concerns, it is also desirable to use a simple infrastructure. Moreover, depending on the dynamic model used to describe the motions of the satellites, different measurements are needed to perform the estimation. The goal of this chapter is to study various measurement infrastructures and to assess the possibility for an algorithm to estimate the state of the fleet based on these measurements (observability). This study will be performed for various dynamic models. The goal of this chapter is to design one measurement architecture, simple enough and still sufficient to make the state of the fleet observable.

2.2 Frames, States, and Measurements

Different dynamic models and measurement infrastructures are analyzed throughout this chapter. For each of these scenarios, an appropriate frame is chosen. In this frame, the state of the fleet (satellites positions, velocities) is also defined. This section presents all the frames, states and measurement infrastructures used throughout this chapter.

2.2.1 Frames

Three frames are considered. Frame A is the absolute frame defined with origin at the Earth center, and axes pointing at stars. This frame is depicted on Fig. 2-1.

Two types of relative frames are considered. Frame R3 is the relative frame defined by Satellites 1, 2 and 3, as depicted on Fig. 2-2. The origin is defined as being Satellite 1's mass center. The x-axis is defined as passing through Satellite 2, with positive x-coordinate. The y-axis is such that the (x, y) plane contains Satellite 3, with positive y-coordinate. Finally, the z-axis is chosen so as to complete a right-hand frame.

As will be seen later, in this frame, the orientations of the spacecraft cannot be estimated. Thus, a second type of relative frame, Frame R1, is defined. It is represented on Fig. 2-3. This frame is defined by one spacecraft only. The origin is this satellite's mass center. The three axes are defined by the spacecraft's orientation: three orientations chosen on the satellite's body define the frame's axes. Fig. 2-4 also represents Frame R1, but zoomed in.

2.2.2 States

Ideally, the goal of the estimation algorithms would be to estimate the "complete" state of the fleet, consisting of

- *Absolute Position* of every satellite in the fleet defined in Frame A;

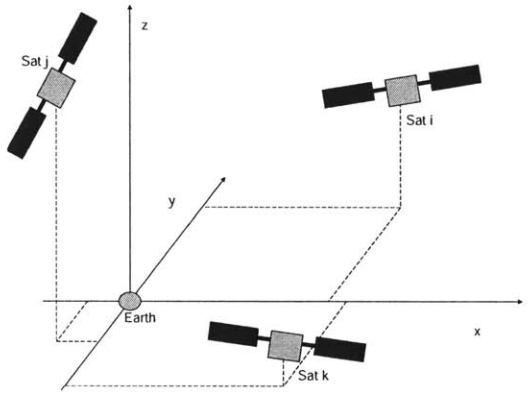

Figure 2-1: Frame A Representation

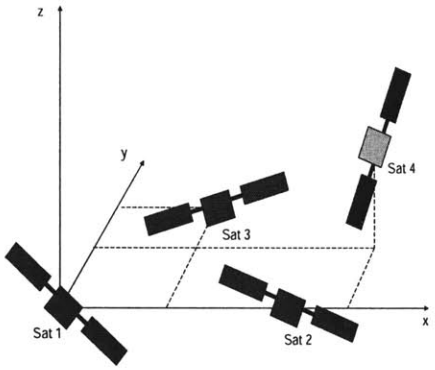

Figure 2-2: Frame R3 representation

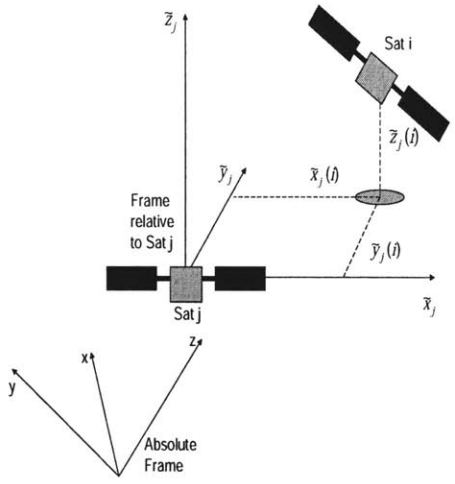

Figure 2-3: Frame R1 representation for Satellite j

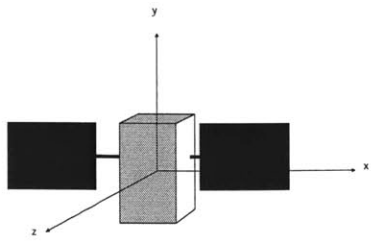

Figure 2-4: Frame R1 Representation, zoomed

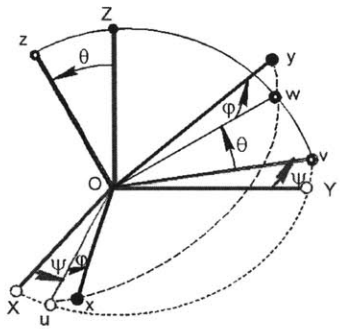

Figure 2-5: Euler Angles Representation: (X, Y, Z) represents Frame A, (x, y, z) Frame R1 based on the considered satellite

- *Absolute Orientation* of every satellite in the fleet. The orientation of a spacecraft is defined by the Euler angles ψ, θ, ϕ of this satellite in Frame A, as depicted on Fig. 2-5.

- *Spacecraft Time Offsets.* In some scenarios, the satellite clocks are assumed to be offset with respect to each other. The time offset t_i of each satellite is defined with respect to the Master, Satellite 1. The time offset δt_{ij} between two satellites is defined by $\delta t_{ij} = t_i - t_j$. In the scenarios where the time offset is considered, it is assumed that a synchronization mechanism (e.g. a flash of light) synchronizes the spacecrafts computations.

The total state vector, based on these elements, is

$$\mathbf{x} = \begin{bmatrix} x_1 & y_1 & z_1 & \psi_1 & \theta_1 & \phi_1 & t_1 & \dots & x_n & y_n & z_n & \psi_n & \theta_n & \phi_n & t_n \end{bmatrix}^T$$

This state is referred to as State 1. However, in most scenarios, it is not possible to estimate all these variables. Thus, other states are defined as versions of this fundamental state.

State 2 is defined as being as state 1, but without considering the time offsets

$$\mathbf{x} = \begin{bmatrix} x_1 & y_1 & z_1 & \psi_1 & \theta_1 & \phi_1 & \dots & x_n & y_n & z_n & \psi_n & \theta_n & \phi_n \end{bmatrix}^T$$

State 3 is defined by the positions, orientations and time offsets of all satellites in Frame R1 based on Satellite 1. Defining:

$$\tilde{\mathbf{x}}_1(2) = \begin{bmatrix} \tilde{x}_1(2) \\ \tilde{y}_1(2) \\ \tilde{z}_1(2) \\ \tilde{\psi}_1(2) \\ \tilde{\theta}_1(2) \\ \tilde{\phi}_1(2) \\ \delta t_{21} \end{bmatrix},$$

where the coordinates $\tilde{x}_1(i), \tilde{y}_1(i), \tilde{z}_1(i)$ are given by

$$\begin{bmatrix} \tilde{x}_1(i) \\ \tilde{y}_1(i) \\ \tilde{z}_1(i) \end{bmatrix} = P_1 \begin{bmatrix} x_i - x_1 \\ y_i - y_1 \\ z_i - z_1 \end{bmatrix},$$

P_1 being the rotation matrix describing the orientation of satellite 1. To make it simpler, cos is replaced by c and sin by s. For instance $c\theta = \cos(\theta)$. With this notation

$$P_1 = \begin{bmatrix} c\theta_1 c\psi_1 & c\theta_1 s\psi_1 & -s\theta_1 \\ -c\phi_1 s\psi_1 + s\phi_1 s\theta_1 c\psi_1 & c\phi_1 c\psi_1 + s\phi_1 s\theta_1 s\psi_1 & s\phi_1 c\theta_1 \\ s\phi_1 s\psi_1 + c\phi_1 s\theta_1 c\psi_1 & -s\phi_1 c\psi_1 + c\phi_1 s\theta_1 s\psi_1 & c\phi_1 c\theta_1 \end{bmatrix},$$

ψ, θ, ϕ being the Euler angles describing the orientation of Satellite 1 in Frame A. With these definitions, State 3 is defined by

$$\mathbf{x} = \begin{bmatrix} \tilde{\mathbf{x}}_1(2) \\ \vdots \\ \tilde{\mathbf{x}}_1(n) \end{bmatrix}$$

State 4 is defined from State 3 by not considering the time offsets. Defining

$$\tilde{\mathbf{x}}'_1(2) = \begin{bmatrix} \tilde{x}_1(2) \\ \tilde{y}_1(2) \\ \tilde{z}_1(2) \\ \tilde{\psi}_1(2) \\ \tilde{\theta}_1(2) \\ \tilde{\phi}_1(2) \\ \delta t_{21} \end{bmatrix},$$

State 4 is given by

$$\mathbf{x} = \begin{bmatrix} \tilde{\mathbf{x}}'_1(2) \\ \vdots \\ \tilde{\mathbf{x}}'_1(n) \end{bmatrix}$$

State 5 is defined as State 4, but without the orientations

$$\mathbf{x} = \begin{bmatrix} \tilde{x}_1(2) & \tilde{y}_1(2) & \tilde{z}_1(2) & \dots & \tilde{x}_1(n) & \tilde{y}_1(n) & \tilde{z}_1(n) \end{bmatrix}^T$$

2.2.3 Measurements

In this chapter, different measurement infrastructures and models are investigated.

Two satellites models are considered: satellites are considered either as point masses, or the position of the transmitters and receivers onboard the spacecrafts are considered. When the spacecrafts are considered point masses, two types of

measurements are investigated: one-way measurements, and two-way measurement. For a two-ways measurement, a beam is sent by a satellite, reaches another satellite, and is sent back to the first satellite. With this measurement model, the following measurement equations hold

- *Crosslink Range Measurement*: measurement of the distance between two satellites. The crosslink range measurement between Satellite i and Satellite j is given by

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} + v_{ij}^r, \ \forall i, j, \tag{2.1}$$

  where the coordinates of each satellite x_k, y_k, z_k are given in the considered frame (this frame depends on the scenario under consideration; see each section). v_{ij}^r is the range measurement noise, a Gaussian white noise of covariance R^r.

- *Elevation* of each satellite as seen from every other: it consists of the angle corresponding to the altitude of a satellite as seen from another (see Fig. 2-6) . For every pair of satellites i and j, the elevation $\theta_j(i)$ of satellite i as seen from j is given by the following two equations

$$\begin{aligned} \tan(\theta_j(i)) &= \frac{\tilde{z}_j(i)}{\sqrt{(\tilde{x}_j(i))^2 + (\tilde{y}_j(i))^2}} \\ \sin(\theta_j(i)) &= \frac{\tilde{z}_j(i)}{\sqrt{(\tilde{x}_j(i))^2 + (\tilde{y}_j(i))^2 + (\tilde{z}_j(i))^2}} \end{aligned} \tag{2.2}$$

  A measurement noise is added to this measurement $\theta_j(i)^{meas} = \theta_j(i) + v_{ij}^{th}$, where v_{ij}^{th} is a Gaussian white noise of covariance R^{th}. In Equation 2.2, the coordinates with a tilde $(\tilde{x}_j(i), \tilde{y}_j(i), \tilde{z}_j(i))$ are the coordinates of Satellite i in Frame R1 defined by Satellite j's position and orientation. They are computed in a similar way as for State 4. P_j is the rotation matrix describing the orientation of Satellite j[1]. Each satellite is also assumed to have an on-board absolute attitude sensor, like a star tracker, so that it can measure its absolute attitude.

[1]The effect of this rotation matrix on the observability is studied in Section 2.4

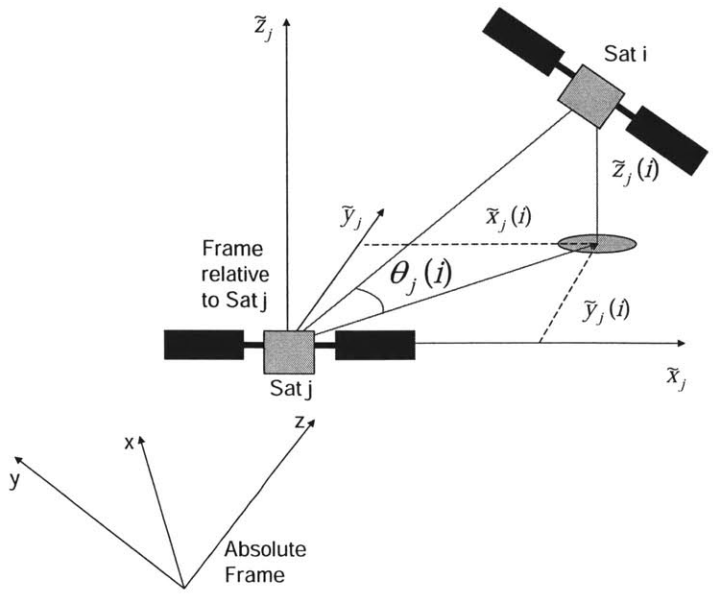

Figure 2-6: Elevation Measurement. The absolute frame is also represented on this illustration

- *Range to Tracking Station Measurement* The range measurement from satellite i to tracking station j is given by

$$\rho_{ij} = \sqrt{(x_i - X_j)^2 + (y_i - Y_j)^2 + (z_i - Z_j)^2} + v_{ij}^{track}, \tag{2.3}$$

  where X_j, Y_j, Z_j are the coordinates of the tracking station (assumed known) and v_{ij}^{track} is the measurement noise, a Gaussian white noise of covariance R^{track}.

Another model considers the position of the transmitters and receivers onboard the spacecrafts. In order to recover the elevation information, each spacecraft carries one transmitter and two receivers. In this study, these devices are assumed to have complete field of view. The receivers are located symmetrically around the (x, y) plane defined in the relative frame R1. In other words, the receivers have the same (x, y) coordinates, but opposite z coordinate. Because of these locations, the range measurements give both a distance information, and an elevation information. The situation considered in this model is depicted on Fig. 2-7.

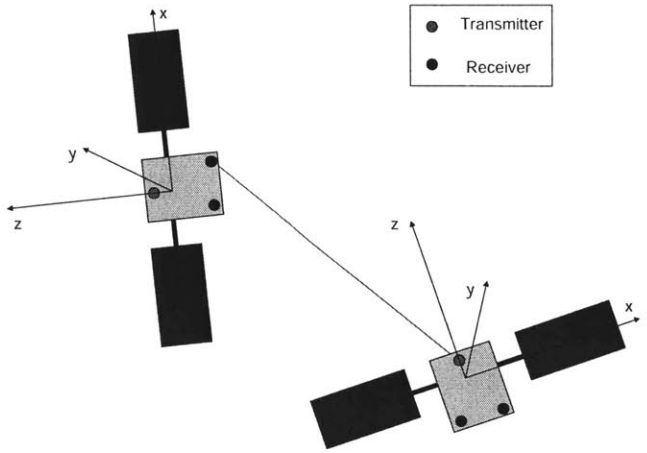

Figure 2-7: Position of the Sensors on the Satellites

The equation corresponding to a two-ways range measurement from the transmitter of Satellite i to the receiver k ($k = 1, 2$) of Satellite j is

$$r_{ij}(k) = \| \left(\mathbf{x}_i + P_i^{-1} \tilde{\mathbf{x}}_{rec,i}(k) \right) - \left(\mathbf{x}_j + P_j^{-1} \tilde{\mathbf{x}}_{trans,j} \right) \| + v_{ij}^r, k = 1, 2 \tag{2.4}$$

where $\mathbf{x}_l = \begin{bmatrix} x_1^l & \dots & x_n^l \end{bmatrix}^T$ is the position vector of Satellite l in Frame A, P_l is the rotation matrix describing Satellite l's orientation, and $\tilde{X}_{rec,l}(k)$ is the position vector of the receiver number k onboard Satellite l, in the frame R1 attached to Satellite l. v_{ij}^r is the measurement noise, a Gaussian white noise of covariance R^r.

Note that the position of the transmitter/receivers onboard the satellite is not considered for the range-to-tracking station measurement. Since the satellite is usually far from the tracking stations, the distance between the sensors and the mass center is neglected as compared with their distance to the tracking station.

The second model considered for the measurements is the one-way measurement model. In this model, one spacecraft sends a beam to another spacecraft. Upon recept, this satellite can compute the distance between the two spacecrafts. However, in this case, the range measurement is biased by the amount of time offset between

the two spacecrafts, according to

$$r_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} + c(t_j - t_i) = d_{ij} + c(t_j - t_i) \qquad (2.5)$$

To summarize, two options are considered with regard to the spacecraft: the spacecrafts are either considered point masses, or the position of the transmitters and receivers onboard the spacecrafts are considered. Two types of range measurements are considered: One-Way and Two-Way range measurements. Finally, three types of measurements are considered

- range-to-satellite measurement
- elevation measurement
- range-to-tracking station measurement

The following measurements scenarios are used in this chapter:

- The spacecrafts are considered point masses
  - The measurements are two-ways
    - *Scenario 1* The spacecrafts are considered point masses. The measurements are two-ways. Range-to-satellite measurements only are considered.
    - *Scenario 2* The spacecrafts are considered point masses. The measurements are two-ways. Range-to-satellite, elevation and absolute attitude measurements are considered.
    - *Scenario 3* The spacecrafts are considered point masses. The measurements are two-ways. Range-to-satellite, elevation and range-to-tracking station measurements are considered.
    - *Scenario 4* The spacecrafts are considered point masses. The measurements are two-ways. Each satellite has Range-to-satellite and 3 range-to-tracking station measurements.
  - The measurements are one-way.
    - *Scenario 5* The spacecrafts are considered point masses. The measurements are one-way. Range-to-satellite and elevation measurements are considered.

- The spacecrafts are not considered point masses. The measurements are two-ways.
  - *Scenario 6* The spacecrafts are not considered point masses: the positions of the transmitters and receivers onboard are considered. The measurements are two-ways. Range-to-satellite measurements only are considered.
  - *Scenario 7* The spacecrafts are not considered point masses: the positions of the transmitters and receivers onboard are considered. The measurements are two-ways. Range-to-satellite and one range-to-tracking station measurements are considered.
  - *Scenario 8* The spacecrafts are not considered point masses: the positions of the transmitters and receivers onboard are considered. The measurements are two-ways. Range-to-satellite and three range-to-tracking station measurements are considered.

2.3 Methods for the Observability Study

This section presents the methods used to assess the observability of the system for a given scenario (measurement infrastructure, state and frame definition).

2.3.1 "Local" and "Static" Observability: Definitions

When estimating the state of a system based on observations, a natural question is to know whether these measurements are sufficient to perform this estimation. This property, known as the "Observability" property, can be determined for a given system. However, one has first to distinguish several definitions of this property.

A system of state x is considered, evolving in time according to the dynamic equation

$$x_{k+1} = f(x_k) + w_k,$$

w being the process noise, of covariance Q. Measurements are performed on this

system, according to equation

$$y_k = h(x_k) + v_k,$$

where y is the measurement vector, h the measurement function, and v the measurement noise, of covariance R.

Local Observability

This chapter aims at determining the observability of the state of the fleet in various scenarios. However, the classical method to assess the observability of a system is a linear method [10, 11, 12, 13, 14]. It relies on computing the rank of the matrix

$$\begin{bmatrix} H^T & A^T H^T & \dots & (A^T)^{n-1} H^T \end{bmatrix}$$

when the dynamics and measurements equations are linear

$$x_{k+1} = Ax_k + w_k,$$
$$y_k = Hx_k + v_k,$$

In order to use this method in our case, in spite of the nonlinearities in the measurements considered, the measurements will be linearized in the vicinity of various points of the state space, according to equation

$$H = \frac{dh}{dx}|\mathbf{x}_0,$$

where $\mathbf{x}_0$ is a point of the state space. As a result, the observability is determined locally only, in the neighborhood of these points in the state space. More precisely, a system is said to be "locally observable" at some point in the state space if, when the system is known to be in the neighborhood of this point, the state of the system can be determined without ambiguity. In contrast, the "global observability" refers to the capability to determine the state of a system without any *a priori* knowledge.

The study presented in this chapter only considers local observability. This property is enough for our purposes, since the initial estimate is close to the actual state of the system.

Complementary approaches about observability for nonlinear systems can be found at [15, 16, 17, 15] provides a theoretical analysis of the different definitions of the observability of a system. [16] investigates the observability of a particular system with a control-theoretic approach. [17] explores a new technique to determine the observability of a system, based on the null space of the measurement jacobian matrix.

Static and Dynamic Observability

The static observability of a system is the capability to determine the state of this system at any given time step, without waiting for the system to evolve in time. If the sensors are such that altogether they capture the state of all the variables describing the system, then this system is said to be statically observable. However, in some cases, the sensors may not be able to capture the state of the system in one single time step. The evolution of the system in time, as measured by these sensors, may be necessary to determine its state. In this case, the system is said to be dynamically observable. Note that if a system is statically observable, it is *a fortiori* dynamically observable.

2.3.2 Observability Determination Method

In this subsection, the method used to determine the local static observability of a system is presented.

Linear Observability Determination Method

For a linear system, the observability is determined by the singularity of the measurement matrix H: the system is (statically) observable if and only if H is full-rank. To determine the singularity of a matrix, one can use the Singular Value Decomposition Theorem.

Singular Value Decomposition (SVD) Theorem

For any $m \times n$ real matrix H, there exist matrices U of size $m \times n$ and V of size $n \times n$, such that $U^T U = V^T V = I$, and an $n \times n$ diagonal matrix D such that [18, 19]

$$H = UDV^T$$

Moreover, the diagonal elements of D are the singular values of the matrix H. The number of non-zero singular values gives the rank of the matrix H. These singular values being by definition the square roots of the eigenvalues of $H^T H$, the following equations holds

$$\begin{aligned} HV &= UD \\ H^T U &= VD \end{aligned}$$

As a result, the columns of V corresponding to the zero singular values in D are in the null space of H. These columns will give insights into non-observability for some systems.

Method used in this study

Based on these methods, the following computations can be performed to determine the local observability of the systems considered

- Computing the Jacobian H of the (nonlinear) measurement function h at a given point x_0 of the state-space (i.e. a given configuration of the fleet).
- Decompose H by use of the Singular Value Decomposition.
- Conclude about the local observability by considering the rank of H.
- The consideration of the null space of H can also provide insight into the lack of observability.

2.3.3 Position Dilution Of Precision

The Position Dilution Of Precision (PDOP) [20] is a metric that evaluates the impact of the fleet geometry on the accuracy of the estimate, for an observable system. For instance, when two measurements are in directions making a small angle, the accuracy of the estimate in a direction perpendicular to these directions will be low. This effect is captured by the PDOP. The PDOP is computed as follows

$$\begin{aligned} G &= \left(H^T H\right)^{-1} \\ PDOP &= \sqrt{trace(G)}, \end{aligned} \tag{2.6}$$

Note that the PDOP is well defined only for observable systems, since H must be non-singular to invert $H^T H$. The PDOP captures the sensitivity of the different measurements, and the amount of additional information they give. The lower the PDOP is, the better the accuracy of the estimate. In the following sections, the PDOP is computed for observable system, so as to gain insight into the performance one can expect for such systems.

2.3.4 Simulation Parameters

In the simulations performed to determine the observability of the various cases, two typical scenarios were considered [6, 21]

- In the first scenario, the satellites are chosen close to the second Lagrangian point L_2 of the Sun-Earth system.
- In the second scenario, the satellites are in an MMS-like configuration (around 1.2×12 Earth radii).

For each simulation (several simulations are performed for each scenario), the positions of the satellites are chosen randomly in the vicinity of these orbits. The measurement function Jacobian H is then computed. The SVD is applied to H, and finally the PDOP is computed.

In some simulations, the orientation of the satellites was assumed to be known. In this case, this orientation was determined as follows: in the frame R1 defined by Satellite 1's position and orientation, Satellite 2 was rotated by $\pi/2$ around x-axis, and Satellite 3 was rotated by $\pi/2$ around y-axis. Section 2.4.4 justifies *a posteriori* this choice.

2.4 Rigid Body Model

2.4.1 Introduction

In this section, the rigid body model is analyzed. In this model, the assumption is made that the fleet does not have any relative motion: the satellites behave as a rigid body, moving altogether in absolute space. Since the goal of our algorithms is to estimate the relative state of the fleet only, the absolute motion is not considered. The state of the fleet consists of the states of the satellites relatively to each other (the state used is given for each measurement infrastructure). For this model, the dynamic observability is equivalent to the static observability, since the state considered does not change in time. This model, although simplified with respect to the other models presented in the following sections, allow a simpler measurement infrastructure to make the system observable.

Two different measurement infrastructures are investigated: the range only configuration (Measurement Scenario 1), and the range and elevation configuration (Scenario 2). The structure of this section is as follows. The first part investigates the range only configuration. The second part studies the range and elevation configuration. The third part analyzes the effect of the relative orientations of the satellites on the observability. The last part considers a more accurate sensor model, and determines the observability with this model.

2.4.2 Range-Only Configuration

Scenario Description

Measurement Scenario 1 is investigated. The range measurement equations is Equation 2.1. Frame R3 is used.

Conceptual Analysis

In this configuration, if the satellites are considered as point masses, with no orientation, the structure of the fleet can be completely determined from the range measurements. Thus, if State 5 is used, because of the frame definition, it should make the system observable. Now, let us consider State 4, that is including the satellites orientations. Since, in this model, the crosslink range measurements measure the distance between the spacecrafts mass centers, the satellites orientations do not impact these measurements. As a result, these orientations are not observable.

Numerical Study

The numerical study confirms the conceptual analysis. In this simulation, the unknown coordinates are considering State 5: z coordinate for Satellite 2, and x and y coordinates for satellite 3 (since Satellite 1 is at the origin of the frame, and $y_2 = z_2 = z_3 = 0$, from the frame definition). The singular values are all non-zero. The system is observable. The PDOP computed in the simulations is usually between 1 and 10 for three satellites. When the satellites are close to be aligned, the PDOP becomes larger, since the geometry does not allow a good accuracy in the estimate. Fig.2-8 depicts a geometry where three satellites are closer and closer to be aligned. Satellite 2 has coordinates $(10, 0, 0)$, and Satellite 3 $(5, y_3, 0)$, with $y_3 = 1, 10^{-1}, ..., 10^{-5}$. As the three satellites are more and more aligned, the PDOP clearly blows out, as depicted on Fig.2-9. This study emphasizes that some configurations decrease the estimation efficiency.

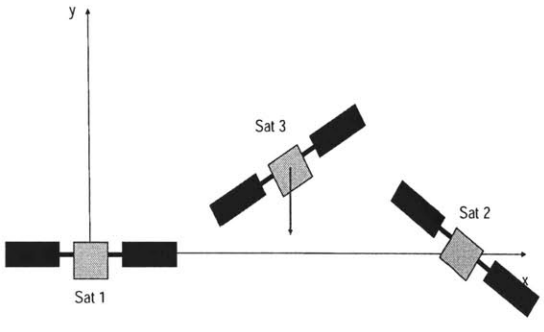

Figure 2-8: Three satellites configuration, where satellite 3 gets closer to be aligned with satellites 1 and 2

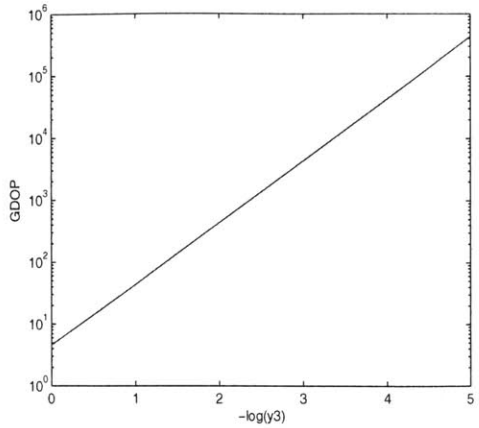

Figure 2-9: Evolution of the PDOP as the three satellites are closer to be aligned

Configuration Advantages

This configuration has an advantage: it is very simple (only range sensors are needed). Moreover, only accurate measurements are used, and the geometry is very good (as long as the satellites are far from being aligned).

Configuration Disadvantages

In this configuration the relative orientations of the satellites (State 4) are not observable. The structure of the fleet can be determined, but not the orientation of each satellite with respect to the others. As a result, every spacecraft can know its distance to every other satellite, but not its position in its own frame of reference.

2.4.3 Range and Elevation Configuration

Scenario Description

In this section, a more sophisticated configuration is considered, aiming at solving the issue mentioned with the previous scenario (non-observability of the satellites relative orientation). Measurement Scenario 2 is considered. The range measurement equation is Equation 2.1. The elevation measurement equation is Equation 2.2. The

frame used is Frame R1 defined by Satellite 1's position and orientation. The state considered is State 4.

Conceptual Analysis

With the addition of the absolute attitude and elevation measurements, the orientations of the satellites are expected to be observable.

Numerical Study

The numerical study shows that, when the fleet consists of three satellites or more, the state is observable. To compute the PDOP, a normalization was done. Indeed, computing the PDOP with both angular and range measurements leads to a huge value (10^5). The reason is that the angular accuracy is to be multiplied by the distance between the satellite to get the actual accuracy in meters. Thus, the range measurements were considered, and the elevation measurements were multiplied by the average distance between the satellites. The PDOP is between 1 and 2. The same study as for the range only configuration was conducted: three satellites almost aligned are considered, with the same parameters. As they become more and more aligned, the PDOP remains below 3 (Fig. 2-10). The elevation measurement has made the orientations observable.

Configuration Advantages

With this configuration, the system is observable, including the orientations of the satellites. Thus, with this sensor infrastructure, all desired information can be estimated.

Configuration Disadvantages

The measurement infrastructure is more complicated than with the range only configuration. Absolute attitude and elevation sensors are needed in this configuration.

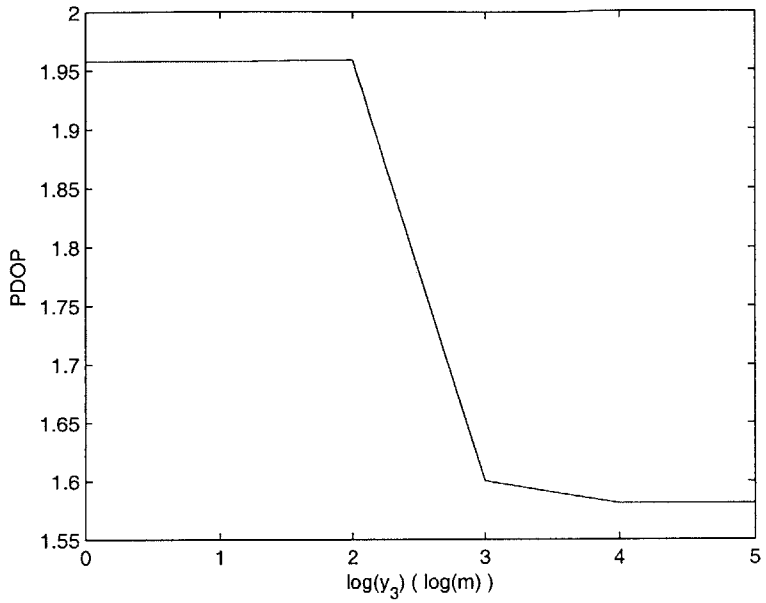

Figure 2-10: Evolution of the PDOP for three satellites almost aligned with range and elevation measurements: the PDOP remains very low thanks to the elevation measurements. To be compared with Fig. 2-9.

2.4.4 Effect of the Satellite Orientation on the Observability

In this section, the effect of the orientations of the satellites on the observability in the range and elevation scenario (Measurement Scenario 2) is studied. In the previous section, the simulations assumed the special orientation described in the methods and definition section. In this section, the effect of this orientation is analyzed. A system of two satellites is considered. The frame considered here is the frame R1 defined by Satellite 1's position and orientation. The effect of the orientation of Satellite 2 on the observability is analyzed. To understand why the orientation of Satellite 2 has an effect on the observability, consider this example: when Satellite 2 has the same orientation as satellite 1, the elevation of satellite 2 as seen from satellite 1 and the elevation of satellite 1 as seen from satellite 2 are redundant information, so that satellite 2 cannot be located, and the system is not observable in these cases. There is no tracking station in this scenario.

An indicator of the observability of the system is $det(H^T H)$. Fig. 2-11 shows

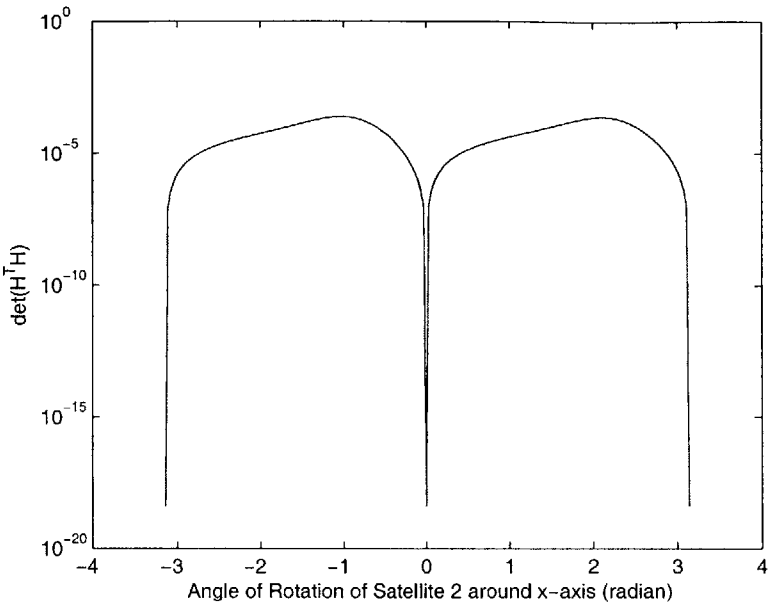

Figure 2-11: Effect of the orientation of a satellite on the observability (in log scale)

the evolution of $det(H^T H)$ as satellite 2's orientation around the x-axis is changed. One can notice that for 0 and π rad rotation, the system is non-observable. This corresponds to the cases when Satellite 1 and 2 have the same orientation. As a result, the system is not observable.

This study shows that in some configurations, the system may not be observable. It also shows that this happens only for a finite number of orientations: in most cases, the system is observable.

2.4.5 Observability when the Sensors are not assumed to be at the Mass Centers

In this section, the effect of assuming that the sensors are not located at the satellites mass centers is analyzed. The model used in this study is the following. Each satellites carries one transmitter, and two receivers. The positions of these sensors are assumed to be known in the satellites body frames R1. These locations are depicted in Fig.2-12. Measurement Scenario 6 is used. State 4 is considered.

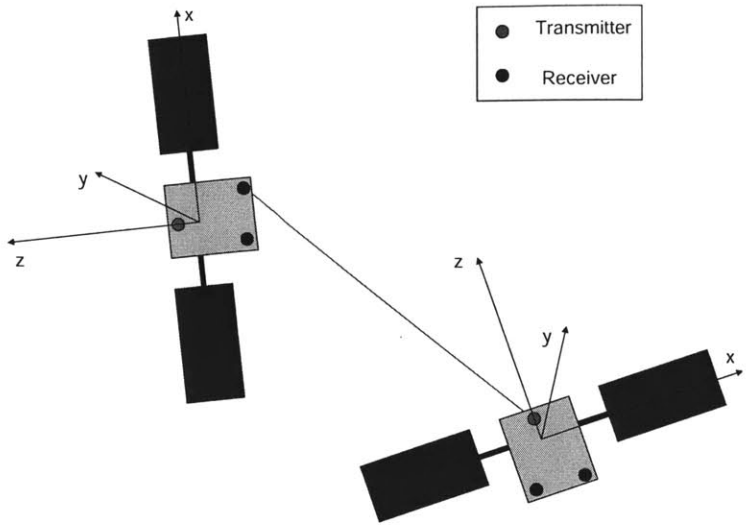

Figure 2-12: Position of the Sensors on the Satellites

In the simulations, the positions of the transmitter and receivers on each satellite was chosen randomly. Each satellite received at each of its receivers a beam from the transmitter of the other satellites, and the measurement matrix was computed from these measurements. The SVD method was used. The simulations showed that the system in this case is observable even without the absolute attitude sensors. Thus, this system is shown to provide the satellites with equivalent information as in the range and elevation model (Measurement Scenario 2).

2.4.6 Conclusion for the Rigid Body Model

Two scenarios were investigated for the Rigid Body Model. The range-only configuration (Measurement Scenario 1) is a very simple configuration, which gives accurate estimates, but for which the orientations of the satellites in the fleet are not observable. Thus, the information necessary to do a maneuver cannot be known in this configuration. The range and elevation configuration (Measurement Scenario 2) is more complicated, but makes the system observable, including the orientations of the satellites. It was also shown that a more realistic model for the sensors, which takes

their location onboard the satellites into account (Measurement Scenario 6), did not change the observability of the system: when the sensors positions are considered, the system -consisting of the satellites positions and orientations- is still observable. Finally, let us note that the Rigid Body Model is a very simplified model. The actual motion of the satellites makes them move with respect to each other. More accurate dynamic models are investigated in the following sections.

2.5 Linear Time-Invariant Model

2.5.1 Introduction

This section presents the observability results obtained for the Linear Time-Invariant (LTI) Model. In this model, the dynamic of the satellites is assumed linear, time-invariant and with satellite independent matrices. Thus, the general LTI form of the motion equation for a satellite i of state X_i (defined in the frame A) is

$$\dot{X}_i = AX_i, \tag{2.7}$$

As a result, by subtracting this equation for Satellites i and j, the following equation describes the relative motion of the fleet

$$\frac{d\left(X_i - X_j\right)}{dt} = A\left(X_i - X_j\right),$$

This equation shows that the relative motion of the fleet does not depend on its absolute state.

Example of a Linear Time-Invariant Dynamic: motion around L_2

Let us give an example of such a dynamic by considering the motion of a satellite in the neighborhood of the Lagrangian L_2 point defined by the Sun-Earth system. The origin of the frame is the L_2 point, the x-axis as passing through the Sun and the Earth, towards the Sun. the y-axis is perpendicular to the x-axis in the plane

of rotation of the Earth-Sun system. Finally, the z-axis is chosen to complete a right-handed reference frame. In this frame, the satellite motion equations are [21]

$$\begin{aligned} \ddot{x} - 2\dot{y} - (1 + c_2)x &= 0 \\ \ddot{y} - 2\dot{x} - (1 - c_2)y &= 0 \\ \ddot{z} + c_2 z &= 0 \end{aligned} \tag{2.8}$$

These equations can be written in the form of Equation 2.7.

For the Linear Time-Invariant Model, two scenarios were investigated: the range-only scenario (Measurement Scenario 1), and the range and elevation scenario (Measurement Scenario 2). Note that these scenarios were found to be statically observable, so that the dynamic model was not used in the observability study. As a result, the study performed for the Rigid Body Model still applies here.

2.5.2 Range-Only Configuration

Scenario Description

Measurement Scenario 1 is considered here. The range measurement equation is given by Equation 2.1. The frame used is Frame R1 defined by Satellite 1.

Conceptual Analysis

In this configuration the system was already studied in Section 2.4. Again, when the satellites are considered point masses, the system is observable. If the orientation of the satellites is added to the state of the system, it is not observable.

Numerical Analysis

The numerical analysis confirms this result. 1000 random configurations have been generated. In any one of these cases, the singular values were all non-zero, making the system statically observable. This static observability implies dynamic observability.

Configuration Advantages

This configuration has an advantage: it is very simple (only range measurements are used). Moreover, only accurate measurements are used, and the geometry is very good (the PDOP computed in the simulations is between 1 and 10).

Configuration Disadvantages

Note, however, that in this configuration the relative orientation of the satellites is not observable. As a result, the satellites do not know in which direction the others lie. The estimation cannot be used for maneuvers, for instance.

2.5.3 Range and Elevation Configuration

Scenario Description

Measurement Scenario 2 is considered. The equation for the range measurement is Equation 2.1. The equation for the elevation measurement is Equation 2.2. Frame R1 attached to Satellite 1 is used.

Conceptual Analysis

By adding the absolute attitude and elevation measurement, the problem mentioned above is solved, making the relative orientation of the satellites observable.

Numerical Analysis

The numerical analysis shows that, for a fleet of three satellites or more, the state is observable. Again, 1000 random configurations have been generated. For each case, the system was found to be observable. The normalized PDOP (see Section 2.4) was found lying between 1 and 2.

Configuration Advantages

In this configuration, the elevation measurements make it possible to know the orientations of the satellites. Thus, the system is observable.

Configuration Disadvantages

The measurement infrastructure is slightly more complicated.

2.5.4 Analysis when the Sensors Positions are considered

The same analysis as in the Rigid Body Model can be conducted when considering the position of the sensors on the satellites. The system is still observable.

2.5.5 Conclusion for the Linear Time-Invariant Model

For the Linear Time-Invariant Model, the results of the Rigid Body Model still apply. Indeed, with the configurations studied, the system is statically observable, so that the dynamic model is not taken into account in the observability study. As a result, as for the Rigid Body Model, the range-only configuration (Measurement Scenario 1) makes the system observable. However, in this configuration, the satellites orientations are not observable. Finally, the range and elevation configuration (Measurement Scenario 2) makes the system observable, included the satellites orientations. It should be noted that the Linear Time-Invariant Model is a more accurate model than the Rigid Body Model, but is still coarse. An accurate dynamic model would show that the relative motion of the satellites depend on their location in absolute space. The following section investigates such a model.

2.6 Inertial Restricted Three Body Dynamic Model

2.6.1 Introduction

In this section, the Inertial Restricted Three Body Model [22, 21] is investigated (centered at body 1 with the mass of all p spacecrafts negligible), in the way presented in [23]. This model considers the motion of three bodies (typically, two planets/stars

and a satellite) when the mass of one of these bodies is assumed negligible. The dynamic model chosen is described by the equation

$$\dot{x}(t) = f(x(t), t) + w(t),\ w(t) \propto N(0, Q),$$

where x is the state of the fleet and w is a Gaussian white noise with covariance defined by the spectral density matrix Q. In this section, the state includes the velocity of the satellites, so that

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}_1^T & \mathbf{v}_1^T & \dots & \mathbf{r}_n^T & \mathbf{v}_n^T \end{bmatrix}^T,$$

where $\mathbf{r}_i$ and $\mathbf{v}_i$ are respectively the position and velocity vectors in the frame A. Then

$$f = \begin{bmatrix} \mathbf{v}_1^T & \mathbf{g}_1^T & \dots & \mathbf{v}_n^T & \mathbf{g}_n^T \end{bmatrix}^T$$

$$\mathbf{g}_i = \ddot{\mathbf{r}}_i = -\mu_1 \frac{\mathbf{r}_i}{r_i^3} - \mu_2 \left(\frac{\mathbf{d}_i}{d_i^3} + \frac{\mathbf{r}_{b_2}}{r_{b_2}^3} \right)$$

Here, μ_j is the gravitational parameter of body j, d_i is the position vector from body 2 to spacecraft i ($\mathbf{d}_i = \mathbf{r}_i - \mathbf{r}_{b_2}$). The Extended Kalman Filter requires to derive f

$$F = \frac{\partial f}{\partial x} = \begin{bmatrix} \mathbf{F}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{F}_n \end{bmatrix} \quad , \quad \mathbf{F}_i = \begin{bmatrix} 0 & I \\ \mathbf{G}_i & 0 \end{bmatrix},$$

where

$$\mathbf{G}_i = \frac{\partial \mathbf{g}_i}{\partial \mathbf{r}_i} = \frac{\mu_1}{r_i^5} \left(3\mathbf{r}_i \mathbf{r}_i^T - r_i^2 I \right) + \frac{\mu_2}{d_i^5} \left(3\mathbf{d}_i \mathbf{d}_i^T - d_i^2 I \right)$$

The dynamic observability condition is given by the rank of

$$\mathbf{O} = \begin{bmatrix} H^T & (HF)^T & (HF^2)^T & \cdots & (HF^{n-1})^T \end{bmatrix}^T$$

The state of the fleet consists of the coordinates of all the spacecrafts in Frame A

$$X = \begin{bmatrix} x_1 & y_1 & z_1 & \dots & x_n & y_n & z_n \end{bmatrix}^T$$

when the orientations of the satellites are considered known, and

$$X = \begin{bmatrix} x_1 & y_1 & z_1 & \theta_1 & \phi_1 & \psi_1 & \dots & x_n & y_n & z_n & \theta_n & \phi_n & \psi_n \end{bmatrix}^T$$

when the orientation θ, ϕ, ψ of every satellite is considered in the state of the fleet.

2.6.2 Range and Elevation Configuration

This section investigates the observability of the system when range and elevation measurements are available at each satellite (Measurement Scenario 2). State 5 is used.

When the frame chosen is Frame A and every satellite receives range, absolute attitude and elevation measurements (Measurement Scenario 2), the system is not observable since any translation of the whole fleet would not change the measurements.

The numerical analysis confirms the non-observability in this case: for a fleet of three satellites, the number of non-zero singular values for a random configuration is 6, whereas the number of degrees of freedom is 9. Thus, the measurement matrix is singular, and the system is not observable.

Moreover, the last three columns of V confirm the reason why the system is not observable. For any random configuration, each of these columns is constituted of the same vector repeated three times

$$u = \begin{bmatrix} a & b & c & a & b & c & a & b & c \end{bmatrix}^T$$

Each set of three lines in this vector corresponds to the coordinates of a satellites.

This means that every translation of the satellites all together lets the measurements unchanged, confirming the intuition presented earlier in this section. The SVD confirms that the system is not observable because any translation would leave the measurements unchanged.

2.6.3 Range, Elevation and one Tracking Station

In this section, a tracking station is added. Measurement Scenario 3 is considered. The equation for the range measurement is Equation 2.1. The equation for the elevation measurement is Equation 2.2. Finally, the equation for the range to tracking station is Equation 2.3. In the previous section, the system was shown to be unobservable because any translation would leave the measurements unchanged. Thus, the effect of adding a tracking station on the observability is investigated. The position of this station is assumed known in the Frame A. It provides measurements of its distance with respect to every satellite. State 2 is considered.

The numerical analysis shows that the system becomes locally statically observable. The singular values of the measurement matrix are all non-zero. Indeed, the measurements from the tracking station remove the translation uncertainty and make the system locally statically observable. The geometrical configurations in which this will not hold can be predicted: when the tracking station and two satellites are aligned.

Note that the measurements from a tracking station are not as accurate as the crosslink range measurements. Thus, the system is observable, but the estimate will be computed from coarse measurements. Moreover, the PDOP is high in this configuration: as seen from the tracking station, which is far from the satellites, any motion in a direction perpendicular to the direction tracking station-satellites has a very small impact on the distance tracking station-satellites. Thus, there is a "blind" direction, and the geometry is not very efficient. The PDOP is approximately 10^3. However, the coarse part of the estimate is the one related with the global translation

of the fleet: the relative state of the fleet is mostly estimated based on the range and elevation measurements, which are accurate. Thus, the relative part of the estimate is expected to be accurate. Moreover, the PDOP is better when the tracking station (or an additional tracking station) is closer to the fleet. For instance, when the fleet is located around the L_2 point, and the first tracking station is on Earth, a second one, located at L_2 makes the PDOP drop to around 2.

2.6.4 Range and three Tracking Stations

In this section, Measurement Scenario 4 is considered. The system with State 2 was shown to be observable in this case. However, the relative state of the satellites - including their relative orientations- is not observable without the tracking stations. Thus, the relative orientations of the satellites is determined by use of the tracking station measurements, which are coarser than the range and elevation measurements. As a result, the estimation is expected to be less accurate in this configuration than in the range, elevation, and one tracking station configuration.

2.6.5 Effect of the Positions of the Sensors on the Satellites

In this section, the effect of considering the locations of the sensors onboard the satellites is analyzed, as was done for the Rigid Body Model.

Range, Elevation and one Tracking Station

The first scenario investigated was range, elevation and one tracking station (Measurement Scenario 7, the sensors are not located at the satellites mass centers, but their locations are known). State 2 is considered. The simulation shows that the system is observable, as it was without this realistic model for the sensors locations.

Range and three Tracking Stations

The second scenario investigated was Measurement Scenario 8. State 2 is considered.

Again, with this scenario and this state definition, the system was found observable.

2.7 Adding Time Offsets in the State

2.7.1 Frame and Measurement Description

In this section, a more realistic model including time offsets between the satellites clock is analyzed. With Frame R1 and the simplified sensor model, two different scenarios have been investigated: one-way and two-way measurements. There is no tracking station. The Measurement Scenario investigated are Scenarios 2 and 5, the sensors are assumed to be located at the mass center. State 3 is considered.

2.7.2 Two-Way Measurements

In this section, the observability for two-way sensors is analyzed. Measurement Scenario 2 is considered. From the two-way assumption, the measurements are not biased by the satellites time offsets. Thus, these time offsets cannot be determined, and the system is clearly not observable. However, one may not need to consider the time offsets if they do not impact the measurements.

The SVD shows, as expected, two null singular values. When looking at the columns of V corresponding to these values, the directions correspond to the time offset coordinates: any change in the satellites time offsets do not change the measurements, as expected since no measurement depends on these time offsets.

2.7.3 One-Way Measurements

In this section, the observability with one-way measurements is analyzed. Measurement Scenario 5 is investigated. By subtracting r_{1i} and r_{i1}, one gets directly satellite i's time offset. Although this costs extra-communication, and degrades the accuracy

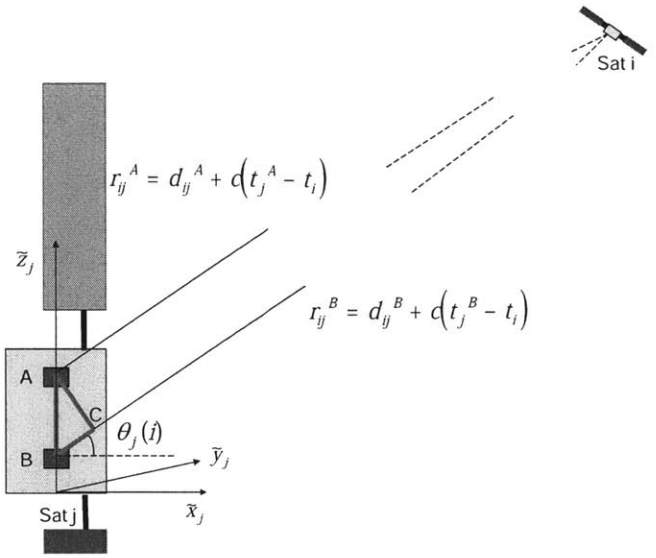

Figure 2-13: Modification of the Elevation Measurement for one-way measurements

of this estimate, the time offsets can now be determined, and the system is expected to be observable in this case.

To understand if the elevation measurement is biased by the time offset between two satellites, one can consider the mechanism used to determine the elevation: two beams are sent by satellite i to satellite j, and the difference in their travel distance gives the elevation, according to Fig. 2-13. Now, if the distance between the two satellites is assumed to be large, the two beams arrive almost parallel at Satellite j. As a result, $\theta_j(i)$ is entirely determined by the triangle ABC. Since this triangle has a right angle and since AB is known, the knowledge of BC is sufficient to know $\theta_j(i)$. But BC is the difference between the length traveled by the two beams. Thus

$$BC = d_{ij}^B - d_{ij}^A = \left(r_{ij}^B - c\left(t_j^B - t_i\right)\right) - \left(r_{ij}^A - c\left(t_j^A - t_i\right)\right),$$

where c is the speed of light. Satellite i time offset cancels out and BC is given by

$$BC = \left(r_{ij}^B - r_{ij}^A\right) - c\left(t_j^B - t_j^A\right)$$

Thus, the elevation measurement will be biased only by the time offset between the two sensors of the same satellite, which can reasonably be assumed to be very low, even when multiplied by the speed of light. For this study, the elevation measurement has been considered not to be modified by the one-way measurements assumption.

Since the difference between two crosslink range measurements provide the time offset of a given satellite, and since the elevation measurements are not modified, the system is expected to be observable in this case.

The numerical analysis confirms this result: there are 8 non zero singular values for a system with three satellites, corresponding to the space coordinates and time offset of Satellites 2 and 3. The system is observable in this case.

2.8 Conclusion

Table 2.1 summarizes the observability study. In this table, the different models studies appear in the first column: Rigid Body Model, Linear Time-Invariant Model, Restricted Three Body Model, and scenarios considering time offsets. The second column defines the Frame considered, with the notation introduced in Section 2.2.1. The third column describes the state, as defined in Section 2.2.2. The next column shows the measurement scenario, as introduced in Section 2.2.3. The fifth column states the number of remaining degrees of freedom, and the last column concludes on the observability of the system thus defined.

The main conclusions from Table 2.1 are

- Rigid Body and Linear Time-Invariant Models. The range only scenario (Measurement Scenario 1) makes the system observable with State 5. However, in this scenario, the orientations of the satellites cannot be determined (State 4 is not observable). In the range and elevation scenario (Measurement Scenario 2), the system is observable with State 4, and the satellite orientations can be determined. Assuming that the sensors are not located at the satellites' mass center (Measurement Scenario 6) does not change the result. The effect of the

Table 2.1: Observability Study Summary for Various Scenarios

Model	Frame	State	Measurement Scenario	Remaining DOF	Observability
Rigid Body	R3	5	1	0	YES
Rigid Body	R1	4	2	0	YES
Rigid Body	R1	4	6	0	YES
LTI	R3	5	1	0	YES
LTI	R1	4	2	0	YES
LTI	R1	4	6	0	YES
Three Body	A	5	2	3	NO
Three Body	A	2	3	0	YES
Three Body	A	2	4	0	YES
Three Body	A	2	7	0	YES
Three Body	A	2	8	0	YES
Time Offset	R1	3	2	(n-1)	NO
Time Offset	R1	3	5	0	YES

satellites orientation was also studied, and emphasized the existence of a finite number of "blind" configurations, in which the system is not observable.

- Restricted Three Body Model. With only range and elevation measurements (Measurement Scenario 2, State 5), the system is not observable, since any translation of the fleet would leave the measurements unchanged. When range, elevation and one tracking station are used (Measurement Scenario 3, State 2), the system is observable. For the scenario using range and three tracking stations (Measurement Scenario 4), the system is also observable for State 2. When the sensors are not assumed to be located at the satellites mass centers (Measurement Scenario 7 and 8), the results do not change.
- Time offsets. When the satellites clocks are assumed to have time offsets (State 3), the system was shown to be observable to one-way measurements (Measurement Scenario 5), but not to two-ways measurements (Measurement Scenario 2).

Table 2.2 summarizes the three main observable systems. These systems are

- Rigid Body and Linear Time-Invariant Models with Range and Elevation mea-

surements (State 4, Measurement Scenario 2). This configuration is simple, but the models are very simplistic.

- Restricted Three Body Model with Range, Elevation and one Tracking Station (State 2, Measurement Scenario 3). This configuration is more complicated but uses accurate measurements and a more realistic model.
- Restricted Three Body Model with Range and three Tracking Stations (State 2, Measurement Scenario 4). This configuration puts more emphasis on ground infrastructure. It is also expected to be less accurate because it relies more on tracking station measurements, which are less accurate.

Table 2.2: Main Observable Systems

Frame	State	Measurements
R1	4	2
A	2	3
A	2	4

The study shows that simple models make the system observable with few measurements (e.g, Rigid Body Model, range measurements only), whereas more realistic models require more measurements to make the system observable (e.g., Restricted Three Body Model not observable with range only measurements). The emphasis can also be put on ground-type measurements, or on onboard sensors, depending on the mission.

For the subsequent studies, the scenario chosen is: one Tracking Station, range and elevation measurements for every pair of satellites, and absolute attitude measurements. This choice makes the system observable, even with the Restricted Three-Body Model and puts more emphasis on the onboard measurement infrastructure.

Chapter 3

From a Centralized Scheme to Decentralized Architectures

3.1 Introduction

3.1.1 Motivation

When facing the estimation problem mentioned in the previous chapter, one method is to gather all the measurements taken by the spacecrafts at the same place, and to process these measurements all together. This approach leads to the "Centralized Algorithm", presented in this chapter. However, the goal in designing navigation algorithms includes:

- Improving the scalability of the communication and computational requirements;
- Improving robustness and reconfigurability (for operation in agile networks);
- Tightly integrating these algorithms with the ranging hardware that will be embedded in the communication network – both to obtain ranging measurements and exchange data between the vehicles.

Thus, as will be emphasized in the following sections, the Centralized Scheme presents a major flaw in that it requires a heavy computational burden to perform this esti-

mation task. As a result, it is desirable to design algorithms which would efficiently split the computational task across the fleet, by use of a Decentralized Scheme. Note that, during the course of this research, the Decentralized Scheme was investigated first in order to distribute the computational burden of the Centralized Algorithm. Later, other issues (e.g. synchronization, communication) arose and were addressed.

3.1.2 The Challenges of Decentralization

This chapter develops this idea by presenting several Decentralized Algorithms. The main technical goal here is to design an estimator architecture to incorporate the available measurement data with minimal impact on the CPU load and communication, and in particular to design distributed estimators that are adaptable to many different missions and control architectures (LEO, highly elliptic orbits, L_2). This filter decentralization is complicated because the nonlinear inter-spacecraft range measurements couple the estimates of the spacecraft states. As a result, every single measurement would need information concerning the state of all the spacecrafts, and an update on both estimates would need to be performed.

The key idea to design a Decentralized Algorithm in spite of this difficulty is to perform a reduced-order estimator based on the Schmidt-Covariance Correction (SCC): as a particular case of the *consider analysis* ([24]), this algorithm takes into account the uncertainty on the knowledge of the state of the other satellites as if it were an additional uncertainty on the measurement itself (see Section 3.4).

Another challenge in designing Decentralized Algorithms is the synchronization level usually required between the satellites to ensure the convergence of the estimate. In this chapter, alternative algorithms requiring various levels of synchronization are presented and discussed. These algorithms are then compared in terms of the following main criteria, defined more precisely in Section 3.10:

- Accuracy: a measure of the error in the estimate;
- Computational Burden: the complexity of the computations to be performed by a given algorithm;

- Synchronization: the extent to which a satellite has to hold up computing to wait for another one to send a crucial piece of information;
- Communication Load: the amount of data transmitted between the spacecrafts.

3.2 The Extended Kalman Filter

This section presents the Extended Kalman Filter as the basis for all the algorithms discussed in this thesis.

3.2.1 Problem Statement

A key capability for many applications is to be able to estimate the state of a system based on measurements performed on this system. The question here is to compute an estimate for this state, while taking into account the accuracy of all the measurements involved. More precisely, one way of representing a dynamic system is the "State-Space Model". It includes a state vector, $\mathbf{x}$, representing the state of the system, which varies according to a dynamic model

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{w}_k, \tag{3.1}$$

where $\mathbf{x}_k$ is the state of the system at time k, f represents the dynamic model, and $\mathbf{w}_k$ is the motion noise at time k, of covariance Q_k. Examples of such a dynamic model were given in Section 2.4, 2.5 and 2.6, where various such models were investigated. The state-space model also includes a measurement model

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \tag{3.2}$$

where $\mathbf{y}_k$ is the measurement vector taken at time k, and $\mathbf{v}_k$ is the measurement noise, of covariance R_k. Examples of such measurements were presented in Section 2.2.3. The problem under consideration is to estimate the state $\mathbf{x}_k$ of the system at

time k based on the measurements up to time k.

3.2.2 The Kalman Filter: An Optimal Solution for the Linear Case

In his seminal paper published in 1960 [25], R.E. Kalman proposed a new filter to estimate the state of a system based on measurements. When the system is described by a linear model -both dynamic and measurement models-, this filter provides an optimal estimate of the state, together with a covariance matrix which evaluates the uncertainty of this estimate. More precisely, if the system is described by the following set of linear equations

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + \mathbf{w}_k, \tag{3.3}$$

$$\mathbf{y}_k = H_k \mathbf{x}_k + \mathbf{v}_k, \tag{3.4}$$

$\mathbf{w}_k$ and $\mathbf{v}_k$ being white noises of respective covariances matrices $E[\mathbf{w}_k \mathbf{w}_l] = Q_k \delta_k^l$ and $E[\mathbf{v}_k \mathbf{v}_l] = R_k \delta_k^l$, with no cross-correlation: $E[\mathbf{v}_k \mathbf{w}_j] = 0$. Then, the Kalman Filter is given by the following equations, computed at each time step k

- Time Update Step

$$\hat{\mathbf{x}}_{k+1}^- = A_k \hat{\mathbf{x}}_k^+ \tag{3.5}$$

$$P_{k+1}^- = A_k P_k^+ A_k^T + Q_k \tag{3.6}$$

- Measurement Update step

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.7}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k \left(\mathbf{y}_k - H_k \hat{\mathbf{x}}_k^- \right) \tag{3.8}$$

$$P_k^+ = (I - K_k H_k) P_k^- \tag{3.9}$$

In these equations, the sign $-$ ($\mathbf{x}_k^-$, P_k^-) denotes the state and covariance after the time update step, and $+$ after the measurement update step. At each time step, both of these sets of equations are computed. The time update step is a prediction of the state from time k to time $k+1$ based on the dynamic model only. The measurement update step processes the measurements at time k to refine the state estimate at this time. A derivation of the Kalman Filter can be found in [26].

3.2.3 The Extended Kalman Filter: Estimating the State of a Nonlinear System

In most applications, the equation describing the system dynamics and the measurements are nonlinear. The Kalman Filter described in the previous section was thus adapted to this context, and the new filter obtained is called the Extended Kalman Filter. This filter is based upon the idea to linearize the dynamics and measurement equations around the estimated state of the system, and to apply the Kalman Filter to the resulting (linear) equations. When the system is described by Equations. 3.1 and 3.2 with f and h not necessarily linear, the Extended Kalman Filter computes, for every time step

- Time Update Step

$$\hat{\mathbf{x}}_{k+1}^- = f(\hat{\mathbf{x}}_k^+) \tag{3.10}$$

$$A_k = \left(\frac{df}{d\mathbf{x}}\right)_{\hat{\mathbf{x}}_k} \tag{3.11}$$

$$P_{k+1}^- = A_k P_k^+ A_k^T + Q_k \tag{3.12}$$

- Measurement Update step

$$H_k = \left(\frac{dh}{d\mathbf{x}}\right)_{\hat{x}_k^-} \tag{3.13}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.14}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k \left(\mathbf{y}_k - h\left(\hat{\mathbf{x}}_k^-\right)\right) \tag{3.15}$$

$$P_k^+ = (I - K_k H_k) P_k^- \tag{3.16}$$

Again, this derivation can be found in [26]. All the algorithms discussed in this thesis are based on the Extended Kalman Filter.

3.3 The Centralized Algorithm

Based on the Extended Kalman Filter, one natural idea to solve the fleet state estimation problem is to gather all the information (measurements) about the system at one single location (one satellite, called the Master), and to perform the Extended Kalman Filter on the global system. The motivation to design such an algorithm is to reach optimality in the estimation process. Indeed, in this algorithm, all information is available at the same location, so that the measurements can be used in the best possible way to estimate the state of the system.

3.3.1 Description of the Algorithm

The Centralized Algorithm is as follows:

1. Each satellite performs measurements;
2. All measurements are sent to the fleet Master;
3. Upon receipt of those measurements, the Master performs the estimation of the whole fleet state, using the Extended Kalman Filter scheme;
4. When completed, the Master sends back to each satellite the estimate of its own position;
5. Back to step 1.

3.3.2 Challenges

The main issues of this algorithm are:

1. Computational Burden. In this algorithm, the whole computation is performed by only one satellite; when the fleet size increases, such a computation becomes hardly possible to perform.
2. Synchronization. The Master has to wait for all the measurements. This increases in particular the vulnerability to network failures.

3.4 Consider Analysis and the Schmidt-Kalman Filter Algorithm

3.4.1 Motivation

As explained earlier, the Centralized Algorithm offers optimality by gathering all information at the same place. However, the number of spacecrafts increasing, it becomes arguably hard to perform the computations required. Moreover, since the Master has to wait for all the fleet before starting to compute, the level of synchronization could become an issue. These reasons make it desirable to decentralize the estimation process by designing efficient distributed algorithms. These arguments are developed in more details in Sections 4.3, 4.4 and 4.6.

Now, decentralizing a decoupled system, for instance based on GPS measurement is a straight-forward process [27], [28]. However, GPS requires constant visibility of the GPS constellation. In space, GPS visibility begins to breakdown at high orbital altitudes (e.g. highly elliptic, GEO, or at L_2). Thus, a measurement augmentation is desired to permit relative navigation through periods of poor visibility and also to improve the accuracy when the GPS constellation is visible [29, 30, 8, 31, 32, 33]. However the local range measurements taken onboard the spacecraft strongly correlate the states of the vehicles, which destroys the block-diagonal nature of the fleet measurement matrix [34, 35] and greatly complicates the process of decentralizing the algorithms [4]. In contrast to the GPS-only estimation scenario that easily decentralizes without loss of accuracy, this estimation problem does not decorrelate

at any level. Since there are significant benefits to dividing the major computation (e.g., estimation, coordination, or control algorithms) across the fleet of vehicles (i.e., robustness, flexibility, and computational speed), Ref. [4] investigated several methods to decentralize the estimation algorithms while retaining as much accuracy as possible.

3.4.2 Consider Analysis

When decentralizing such a coupled system, one has to take into account the uncertainty on the other satellite as an increased uncertainty on the cross-link measurement. This idea is the basis of the *consider analysis* (CA), developed by Biermann [36]. Consider Analysis typically partitions the state into $\mathbf{x}$, which is to be estimated (local state of the satellite), and $\mathbf{y}$, which is to be "considered" (in our case, states of all the other satellites). Considered in this context means that the role of uncertainty in $\mathbf{y}$ is investigated through the a priori covariance P_y that corrupts (increases) the covariance of the measurement R. Consider Analysis also analyzes the effects of errors in other *consider parameters*, such as incorrect a priori covariance for the estimated parameters, incorrect measurement noise, and errors in the transition and measurement matrices. Biermann presents both consider analysis (impact of the un/mis-estimated parameters on a filter's accuracy) and consider filtering (redesign of a filter to reduce impact of unestimated parameters). These last developments are not used in the Schmidt-Kalman Filter (SKF).

3.4.3 Reduced-Order Decentralized Filters

One of the major issues with the Centralized Algorithm is the computational burden it must perform. The reason for this burden is the size of the state to be estimated: since it consists of the whole fleet, the computations involved are heavy. This consideration gave rise to the principle of the reduced-order Decentralized Filters. In such filters, the state to be estimated is only the state of the local spacecraft. The state of the

other satellites in the fleet are used, but not estimated: the estimate for these states is received through inter-satellite communications channels.

As an example of such a reduced-order Decentralized Filter, Ref. [3] introduced the Iterative Cascade Extended Kalman Filter (ICEKF). This filter is used for local ranging augmentation for applications where GPS-only measurements are not sufficient. The ICEKF filter uses an iterative technique that relies on communication between each vehicle in the fleet and continues until a specified level of convergence has been reached. It was shown that ICEKF can incorporate local ranging measurements with GPS levels of accuracy, producing nearly optimal performance. However, Ref. [37] demonstrated that the filter performance can deteriorate when highly accurate local measurements (i.e., more accurate than GPS) were added, and that this performance loss occurs when error/uncertainty in the relative state vectors is not correctly accounted for in the filter. One way to account for this uncertainty in the relative state is to include it in the measurement noise covariance R, which is the approach taken in the *Bump Up R* method

$$R_{\text{bump}} = R + JP_{\text{yy}}J^T \tag{3.17}$$

where J is the measurement matrix for all non-local measurements in the fleet and P_{yy} is the initial covariance matrix for all non-local states in the fleet state vector. Equation 3.17 implies that the measurements now have larger noise covariance, making the measurements less accurate than was initially assumed, which is a more accurate description of the scenario.

3.4.4 Description of the Algorithm

Another approach examined in Ref. [37] is the *Schmidt Kalman Filter* (SKF). This filter also increases the variances in the R matrix, but in contrast to *Bump Up R*, this approach is dynamic and it also accounts for the off-diagonal blocks of the error covariance. This estimator eliminates non-local state information, thereby reducing

the computational load on the processor. This elimination is accomplished by partitioning the measurement and propagation equations

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}_{k+1} = \begin{bmatrix} A_{\mathbf{x}} & 0 \\ 0 & A_{\mathbf{y}} \end{bmatrix}_k \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}_k + \begin{bmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{bmatrix}_k \tag{3.18}$$

$$\mathbf{z}_k = h\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}\right) + \nu_k \tag{3.19}$$

$$P_k = \begin{bmatrix} P_{\mathbf{xx}} & P_{\mathbf{yx}} \\ P_{\mathbf{xy}} & P_{\mathbf{yy}} \end{bmatrix}_k \tag{3.20}$$

where $\mathbf{x}_k$ represents the vector containing the states of interest (called the *local state*, which includes the positions, velocities, and time of the vehicle relative to the fleet origin), and $\mathbf{y}_k$ represents the remaining states (i.e., the positions, velocities, and time of all other vehicles relative to the fleet origin). Using the partitions of Equations 3.18 and 3.19 applied to the general Kalman filter equations, each block of the covariance update equations can be solved for, and the gain for the $\mathbf{y}$ states is then set to zero [38] ($\mathbf{y}$ is not estimated locally instead its estimate is provided by the other satellites). This gives the *Schmidt-Kalman Measurement Update* equations

$$\alpha_k = H_k P^-_{\mathbf{xx}_k} H_k^T + H_k P^-_{\mathbf{xy}_k} J_k^T + J_k P^-_{\mathbf{yx}_k} H_k^T + J_k P^-_{\mathbf{yy}_k} J_k^T + R_k \tag{3.21}$$

$$K_k = (P^-_{\mathbf{xx}_k} H_k^T + P^-_{\mathbf{xy}_k} J^T)\alpha_k^{-1} \tag{3.22}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\left\{\mathbf{z}_k - h\begin{pmatrix} \hat{\mathbf{x}}_k^- \\ \hat{\mathbf{y}}_k^- \end{pmatrix}\right\} \tag{3.23}$$

$$P^+_{\mathbf{xx}_k} = (I - K_k H_k)P^-_{\mathbf{xx}_k} - K_k J_k P^-_{\mathbf{yx}_k} \text{ and } P^+_{\mathbf{yy}_k} = P^-_{\mathbf{yy}_k} \tag{3.24}$$

$$P^+_{\mathbf{xy}_k} = (I - K_k H_k)P^-_{\mathbf{xy}_k} - K_k J_k P^-_{\mathbf{yy}_k} \text{ and } P^+_{\mathbf{yx}_k} = P^{-T}_{\mathbf{xy}_{k+1}} \tag{3.25}$$

where $H = \partial h/\partial \mathbf{x}|_{\hat{\mathbf{x}}_k^-,\, \hat{\mathbf{y}}_k^-}$ and $J = \partial h/\partial \mathbf{y}|_{\hat{\mathbf{x}}_k^-,\, \hat{\mathbf{y}}_k^-}$. The second component is the *Schmidt-Kalman Time Update*

$$\hat{\mathbf{x}}_{k+1}^- = A_{\mathbf{x}_k}\hat{\mathbf{x}}_k^+ \tag{3.26}$$

$$P^-_{\mathbf{x}\mathbf{x}_{k+1}} = A_{\mathbf{x}_k} P^+_{\mathbf{x}\mathbf{x}_k} A^T_{\mathbf{x}_k} + Q_{x_k} \tag{3.27}$$

$$P^-_{\mathbf{x}\mathbf{y}_{k+1}} = A_{\mathbf{x}_k} P^+_{\mathbf{x}\mathbf{y}_k} A^T_{\mathbf{y}_k} \text{ and } P^-_{\mathbf{y}\mathbf{x}_{k+1}} = P^{-T}_{\mathbf{x}\mathbf{y}_{k+1}} \tag{3.28}$$

$$P^-_{\mathbf{y}\mathbf{y}_{k+1}} = A_{\mathbf{y}_k} P^+_{\mathbf{y}\mathbf{y}_k} A^T_{\mathbf{y}_k} + Q_{y_k} \tag{3.29}$$

For the SKF to compute an appropriate amount to increase R, each spacecraft communicates both its local state vector and its local error covariance matrix to the next spacecraft in the fleet (this gives each satellite access to $\hat{y}_k^-$ that appears in Equation 3.23 and in the calculation of H and J). This effective change to R is called the *Schmidt Covariance Correction* (SCC) [5]. The additional error covariance information, which is not transmitted when using the *Bump Up R* method, allows a more appropriate correction, but also requires additional inter-spacecraft communication.

The approach used by the SKF to share specific information between satellites can be applied to other filters, including those using more Centralized Architectures. This is achieved using the Schmidt covariance correction, which allows the various estimation architectures to correctly account for errors in the range measurements *without* having to estimate the states of those vehicles. This can greatly improve the performance and adaptability of the estimation approach, which are important properties for reconfigurable networks. For example, the SKF can also be used to develop a new type of Hierarchic Architecture that is quite different than a traditional hierarchy because the spacecrafts are allowed to communicate with vehicles in their local cluster **and** with vehicles in other clusters (see Chapter 4). This is accomplished by using the Schmidt Covariance Correction to avoid having to increase the estimator size. The SCC enables each spacecraft to range off spacecraft in other clusters (cross-team ranging) without having to estimate their relative states. The SCC essentially provides each spacecraft with a way to receive and correctly implement new information, regardless of where this information is coming from.

In the context of Decentralized Architectures, the Schmidt Kalman Filter is first used to design the Iterative Schmidt Kalman Filter (ISKF, which combines the Iterative Cascade Extended Kalman Filter from Park [3] with the Schmidt-Kalman Filter [39]), depicted on Fig. 3-1. In this algorithm, the following steps are completed

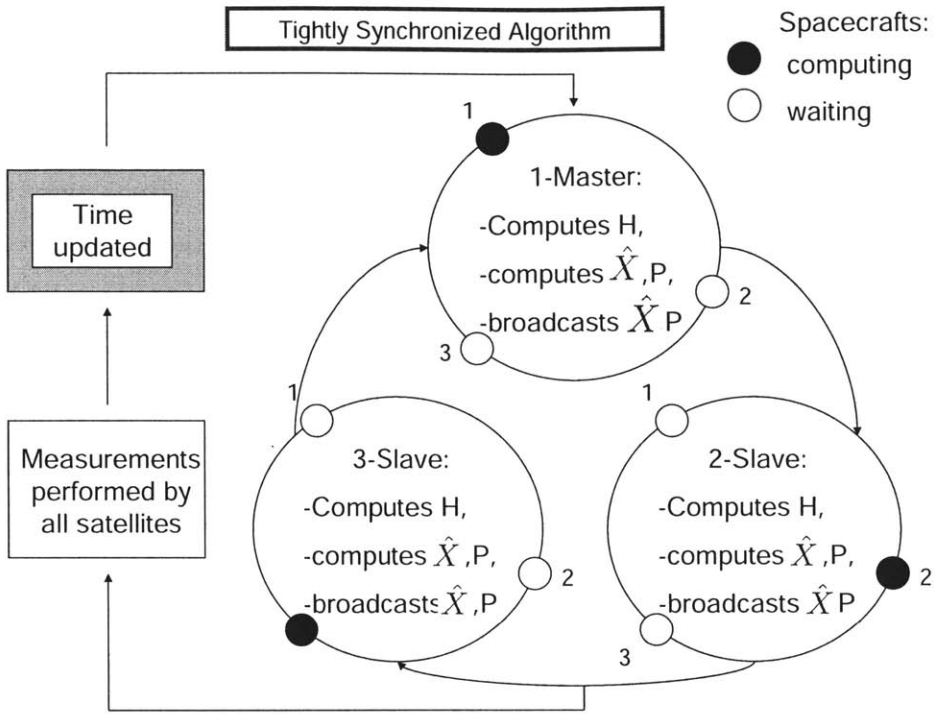

Figure 3-1: Iterative Schmidt-Kalman Filter Algorithm – Tightly Synchronized

1. All satellites take measurements at the same time. Time is then considered "frozen" while a measurement update is done;

2. The first satellite in the fleet does a measurement update, using the global fleet estimate from the previous time step. All other satellites wait for this result;

3. When completed, the updated estimate (state and covariance) is broadcast to all other satellites for future use;

4. The next satellite performs a similar process: update and broadcast;

5. This process is completed around the fleet (or the measurement group), a total of I times (usually, $I = 2$), see discussion in Section 4.6;

6. Each satellite performs a time update step, and the process is repeated from step 1.

3.4.5 Challenges

This algorithm is an approximation of the optimal Centralized Algorithm, since the local filters are reduced order and are not necessarily updated to reflect all measurements taken in the fleet. This leads to a significantly improved distribution of the computation, fleet scalability and robustness, but does degrade performance. Also, it adds complexity to communications and information sharing. This puts limit on the size of the fleet running the Decentralized Filter [4, 5]. Indeed, since the algorithm used requires multiple iterations around the fleet for each measurement update, there is a high level of synchronization – every satellite must wait for the whole loop around the fleet.

Previous work showed the ISKF as a means to reduce the amount of computational burden undergone by each spacecraft [4, 37], by distributing across the fleet an approximative form of the Centralized Filter, and iterating it around the fleet so that to restore the original accuracy of this one. However, when the size of the fleet increases, the synchronization issues increase as well: network failure, or any delay in computation, for instance, would impact on the whole computational process, and possibly stop it. This situation is very problematic, so the goal of more recent work has been to evaluate alternative algorithms that have lower synchronization requirements while maintaining an acceptable level of accuracy, low computational load, and a reasonable amount of communication.

3.5 The Low-Level Synchronization Algorithm

The main concern with the ISKF is its high level of synchronization: at the end of each measurement update, every satellite needs to wait for all others.

3.5.1 Description of the Algorithm

To reduce this synchronization, one idea is for every satellite to perform these measurement updates without sending the result and waiting for the others. At the end of the whole loop, each spacecraft sends the estimation and waits for the others. In other terms, all satellites process in parallel, at the same time. This defines the Low-Level Synchronization Algorithm.

3.5.2 Challenges

This algorithm decreases the synchronization by not requiring a redistribution of the new state and covariance estimates during the measurement update. Synchronization is still required, but this is at a much lower level because, between two consecutive estimation steps, each satellite only has to wait once for the other vehicles. In the original ISKF approach each satellite must wait I (number of iteration in the ISKF) times for each of the other satellites. This gain in synchronization impacts the accuracy, and this effect needs to be analyzed.

3.6 The Fully Asynchronous Algorithm

This algorithm takes a radical way of eliminating the synchronization issues of the previous algorithms. Indeed, no satellite waits for others to computing.

3.6.1 Description of the Algorithm

The fully asynchronous scheme is depicted in Fig. 3-2. Each satellite executes the following algorithm using the available updates from others, but without "waiting" for the information to arrive

1. Takes the measurements;
2. Performs a measurement update – iterated SKF;

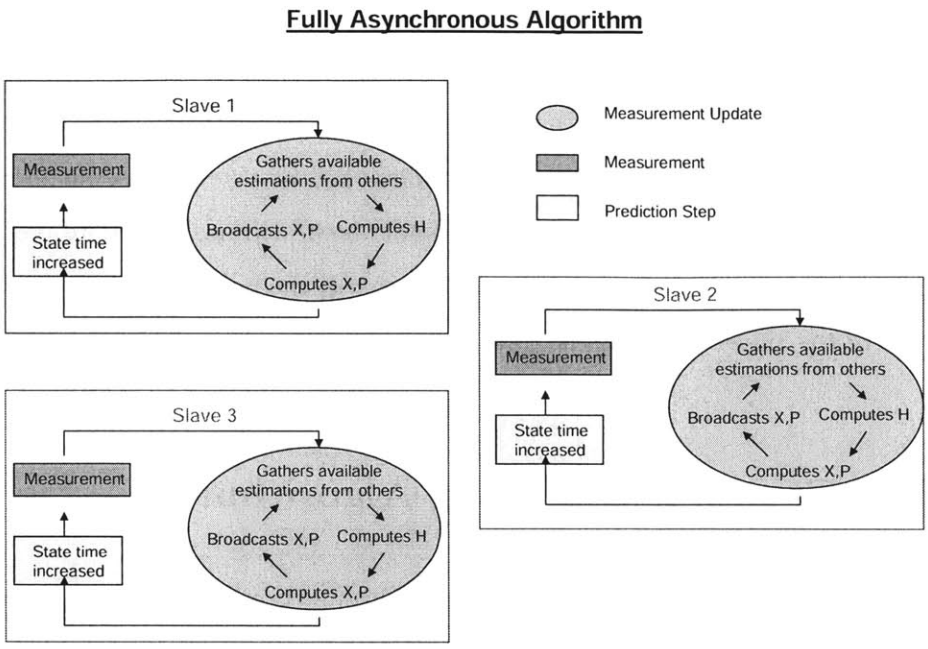

Figure 3-2: Fully Asynchronous Algorithm

3. Broadcasts its updated location estimate $\mathbf{x}_k^+$;
4. Performs a time update to the next time step and broadcasts that estimate $\mathbf{x}_{k+1}^-$;
5. Repeats from step 1.

3.6.2 Challenges

The Fully Asynchronous Algorithm discussed above fixes all of the synchronization issues, but this success is achieved with a potentially large loss of accuracy. The following discusses some ways that have been explored to recover this accuracy:

- Processing sequentially: The primary reason for the loss of accuracy is that the satellites are using information (estimates and covariances) from others at a different time (the measurement taking and updates are not synchronized) than when these measurements were received. The algorithm could be modified to process this data sequentially: propagate our own estimate to the time that the

i^{th} satellite sent its information and include it without adding to the covariance, and do that for all spacecrafts. However, this neglects the cross-correlations between the various satellites, and should be done only if these cross-correlations are low.

- Processing at a relevant time: If now the data are not processed sequentially, one could think about processing these data at the best time, so that to reduce the amount by which the covariances are increased; the best time could be the average of the times at which the data were received.

3.7 The Constant Time-Lag Algorithm

The Fully Asynchronous Algorithm solved the synchronization problem, but potentially sacrifices too much performance to achieve it. An alternative is to slightly modify the ISKF. In particular, if there is a short delay in the computation of one of the satellites, it might be beneficial for the others to wait a bit, so that the additional estimation information arrives. The original ISKF embedded a wait period, but it was potentially quite long, which is problematic. To avoid these long delays, an algorithm that waits for a constant time-lag could be used, which would give the algorithm a better chance to get new update without the risk of waiting for too long. The result strikes a good balance between fully synchronized and asynchronous.

3.7.1 Description of the Algorithm

This algorithm is essentially the same as the Fully Asynchronous Algorithm, but, at the end of a cycle {Measurement Update Step + Prediction Step}, each satellite waits a given amount of time, with the intention that this additional period of time will enable others to finish their loops and broadcast their updates estimates. However, there is no explicit waiting for information from other vehicles – once the time is up, the algorithm moves on.

3.7.2 Challenges

The main issue with this algorithm is the selection of the time-lag value. Indeed, this value must be selected so that the current estimate is not too corrupted by waiting too long, but each satellite still waits long enough to significantly increase the probability that information from another satellites will arrive. Thus, this parameter is hard to chose.

3.8 The Partially Fleet Synchronized Algorithm

A further strategy of reducing the risk of waiting for a long time for information from the other vehicles in the fleet is to implement an algorithm where every satellite waits for fewer spacecrafts. In particular, if it is possible to pick some of the satellites that are predicted to have the largest impact on our estimate, then these should be waited for, but the others can be by-passed if not received in time.

3.8.1 Description of the Algorithm

This algorithm uses the scheme of the Fully Asynchronous Algorithm, but waiting for some specific satellites of the fleet. More precisely, each satellite

1. Takes the measurements;
2. Waits for the satellites known to have great impact on its estimation;
3. Performs a measurement update – iterated SKF;
4. Broadcasts its updated location estimate x_k^+ and covariance P_{xx};
5. Waits again for the relevant other satellites, and does again in loop I times the measurement update;
6. Performs a time update to the next time step and broadcasts that estimate x_{k+1}^- and covariance P_{xx};
7. Repeats from step 1.

Each satellite only has to wait for part of the fleet: the satellites whose estimation have the greatest impact on its own estimate. To determine this part of the fleet, the following section presents an analysis of the sensitivity of a satellite's estimate with respect to the information (state and covariance) of a second satellite.

3.8.2 Challenges

In this algorithm, the synchronization is reduced, and the important information only is waited for. However, this strategy raises one major challenge in that it requires the capability to determine the important satellites for each spacecraft. The following section addresses this issue by analyzing the impact of the knowledge about one satellite on the estimate of another satellite. Now, again, a trade-off rises between the accuracy obtained by waiting for many satellites, and the synchronization constraint associated with it.

3.8.3 Sensitivity Study

A key part of the Partially Asynchronous Algorithm is to list the important satellites whose estimations improvements will have the greatest impact on its own estimate. This can be predicted using the sensitivity of the current state and covariance information with respect to knowledge of the other vehicles. The covariance analysis starts with the formula

$$\begin{aligned} \hat{P}_{xx} &= (I-KH)\,\hat{P}_{xx}^{-}\,(I-KH)^{T} + K\left(R + J\hat{P}_{yy}^{-}J^{T}\right)K^{T} \\ &\quad - KJ\hat{P}_{yx}^{-}\,(I-KH)^{T} - (I-KH)\,\hat{P}_{xy}^{-}\,(KJ)^{T} \end{aligned} \tag{3.30}$$

which simply gives the estimate of the updated covariance as a function of the filter gain and the prior knowledge of all vehicle locations. The Kalman gain is computed

by optimizing $\mathtt{tr}\left(\widehat{P}_{xx}\right)$ with respect to K, which gives

$$K = \left(\widehat{P}_{xx}^{-} H^T + \widehat{P}_{xy}^{-} J^T\right) \alpha^{-1} \tag{3.31}$$

where, as before,

$$\alpha = R + H\widehat{P}_{xx}^{-} H^T + J\widehat{P}_{yy}^{-} J^T + H\widehat{P}_{xy}^{-} J^T + J\widehat{P}_{yx}^{-} H^T \tag{3.32}$$

The SKF state update is computed as (see Equation 3.23)

$$\hat{\mathbf{x}}_k^{+} = \hat{\mathbf{x}}_k^{-} + K_k \left\{ \mathbf{z}_k - h \begin{pmatrix} \hat{\mathbf{x}}_k^{-} \\ \hat{\mathbf{y}}_k^{-} \end{pmatrix} \right\} \tag{3.33}$$

where $\hat{\mathbf{x}}_k^{-/+}$ represent the state estimates (before and after current update) of the satellite performing the analysis, and $\hat{\mathbf{y}}_k^{-}$ the state estimate of all other satellites.

Sensitivity of P_{xx} with respect to P_{yy}

From the expression of the covariance given in previous section, we want to estimate

$$\frac{\partial \mathtt{tr}\left(\widehat{P}_{xx}\right)}{\partial \widehat{P}_{yy}}$$

From Equation 3.30, one gets

$$\frac{\partial \mathtt{tr}\left(\widehat{P}_{xx}\right)}{\partial \widehat{P}_{yy}} = \frac{\partial \mathtt{tr}\left(KJ\widehat{P}_{yy}J^T K^T\right)}{\partial \widehat{P}_{yy}} = \frac{\partial \mathtt{tr}\left(\widehat{P}_{yy}\left(KJ\right)^T\left(KJ\right)\right)}{\partial \widehat{P}_{yy}}$$

and with $\frac{\partial tr(AB)}{\partial A} = B^T$, one finally gets

$$\frac{\partial \mathtt{tr}\left(\widehat{P}_{xx}\right)}{\partial \widehat{P}_{yy}} = \left(KJ\right)^T\left(KJ\right) \tag{3.34}$$

Sensitivity of $\hat{\mathbf{x}}_k^{-}$ with respect to $\hat{\mathbf{y}}_k^{-}$

In the equation of the state update, $\hat{\mathbf{y}}_k^{-}$ appears explicitly in the measurement pre-

diction $h\begin{pmatrix}\hat{\mathbf{x}}_k^- \\ \hat{\mathbf{y}}_k^-\end{pmatrix}$, but is also implicitly embedded in the computation of K_k through that of $H = \left(\frac{\partial h}{\partial x}\right)_{\hat{x}_k^-,\hat{y}_k^-}$ and $J = \left(\frac{\partial h}{\partial y}\right)_{\hat{x}_k^-,\hat{y}_k^-}$. Thus, the nonlinearity makes this analysis complicated, and a precise computation must also trace the impact of $\hat{y}_k^-$ changes on K as well. However, some insights can be developed from the primary effect of $\hat{y}_k^-$ on $\hat{x}_k^-$ by considering K_k frozen

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\left\{\mathbf{z}_k - h\begin{pmatrix}\hat{\mathbf{x}}_k^- \\ \hat{\mathbf{y}}_k^-\end{pmatrix}\right\}$$

Given this equation, it is clear that

$$\left.\frac{\partial \hat{\mathbf{x}}_k^+}{\partial \hat{\mathbf{y}}_k^-}\right|_{K_k} = -K_k J_k \tag{3.35}$$

Thus each satellite can use the computation of KJ to evaluate the impact of a better knowledge about another satellite on its own estimation. More precisely, given the fleet estimation at any instant, the satellite computes KJ. Then, for a small variation in the estimation $\delta\widehat{\mathbf{y}^p}_k^-$ of the satellite p

$$\delta\hat{\mathbf{x}}_k^+ \approx KJ\delta\widehat{\mathbf{y}^p}_k^- = \begin{bmatrix} KJ_1, \cdots, KJ_p, \cdots, KJ_{n-1}\end{bmatrix}\begin{pmatrix}0 \\ \vdots \\ \delta\widehat{\mathbf{y}^p}_k^- \\ \vdots \\ 0\end{pmatrix} = KJ_p\ \delta\widehat{\mathbf{y}^p}_k^-$$

which shows, that by comparing the block matrices KJ_p corresponding to the other satellites of the fleet, each satellite can determine the ones that are most important to it, and then only wait for these satellites.

3.9 The Pairwise Algorithm

Another strategy to develop an algorithm more asynchronous than the Centralized and Decentralized Algorithm is to process the data sequentially, so that the updates are always done pairwise. In Ref. [40] – a study of multi-robot localization problems – it is shown that the Centralized Extended Kalman Filter can be implemented in a distributed fashion, while remaining optimal. To compare this algorithm with the other algorithms presented here, a version of the *fully asynchronous* approach that is adapted to their scenario was considered. The result is that their scheme requires extra-communication to ensure that **all** other vehicles are informed of a measurement update between two vehicles. For example, if robot 2 and 3 meet, exchange data, and take ranging measurements between them, then, in their optimal approach, all other robots in the fleet need to know the measurement and the old estimates of robots 2 and 3.

While the Schmidt Kalman Filter Algorithm is not an equivalent but distributed version of the Centralized Algorithm, as the Pairwise Algorithm, it avoids this extra communication and synchronization step. Note that for a large group of vehicles ($n \gg 1$) the minimum number of direct communication links can be a source of problem for their implementation strategy. Indeed, for each of the $n(n-1)/2$ pairwise measurements, the whole fleet has to be informed about the new piece of information that has arrived. In contrast, in our algorithm, each satellite uses all its measurements at the same time, and broadcasts its estimation only after the measurement update, and without enforcing the requirement that it be used by all others.

There is a further important difference for cases when there are multiple measurements at the same time. Specifically, the algorithm in Ref. [40] gives an optimal estimation approach from one measurement between two robots, but when used sequentially on many measurements, the resulting estimation is not necessarily optimal (a succession of optimal processes is not always optimal). Indeed, the Fully Asynchronous Algorithm takes advantage of the cross-correlations between the different

measurements (for instance through the off (block) diagonal terms of $HP_{xx}H^T$, which reflects the extra knowledge that $\mathbf{x}$ is involved in all the measurements), and this is not the case with a sequential implementation. This algorithm will not be included in the comparison presented thereafter.

3.10 Qualitative Comparison

3.10.1 Problem Statement

The Iterated SKF (ISKF) Algorithm presented in Section 3.4 uses a sequential update approach which requires that each vehicle wait in turn to update its states/covariance. This scheme requires both a large amount of communication and a tight synchronization between the satellites. This may become a problem if any of the spacecraft is delayed for any reason, and such a risk increases when the number of vehicles in the fleet becomes large. Thus two kinds of synchronization are distinguished:

- *Fleet Synchronization* refers to the number of times (or amount of time) that a satellite must wait for information (e.g., its estimation and covariance) from another satellite in the fleet during a measurement update step;
- *Time Synchronization* refers to the number of times (or amount of time) that a satellite must wait for other satellites to finish the time-update step.

Together, Time and Fleet Synchronizations capture the extent to which a satellite must hold up its own calculations waiting for information from another vehicle. The areas in which the efforts to compare the algorithms were focused are

- Accuracy. The estimation process is desired to reach the best accuracy, both in "normal use", and in case of failure (in the network for instance);
- Computational Burden. A great accuracy would be of no use if it takes one hour to compute it. Thus, the computational load is also a factor to be minimized;
- Synchronization. Waiting for another satellite is considered a flaw, since it could force a satellite to wait possibly for a long time;

- Communication Load. The amount of data exchanged through the communication channel to perform the estimation is to be reduced.

Several algorithms have been investigated, each of which one fixes some flaws in the areas listed above. Typically, fixing one issue may impact another factor, and there is no perfect algorithm with respect to the criteria listed. As a result, it is desirable to know for which criteria each algorithm performs well: thus, a metric was designed to evaluate each algorithm in the various areas listed. The following lists the algorithms explored.

3.10.2 Evaluation Metrics

It is clear from the preceding that this study is a multiple-objective design problem: the goal is to design an algorithm that at the same time improves accuracy, is faster, uses less communication, and shows lower sensitivity to network failures. The goal of this section is to present two metrics that can be used to capture the "global efficiency" of a given algorithm

- The first metric is called "performance" and captures the accuracy of the algorithm (discrepancy between computed estimation and actual state);
- The second metric, called the "penalty", is designed to capture the hidden flaws/issues embedded in the algorithm, e.g., the synchronization constraint, the communication burden, and the computation load.

Both metrics are designed to be lower for a better algorithm. For a given algorithm, it will be possible to draw on a frame (penalty, performance) a cross corresponding to each algorithm. However, note that depending on the application, the relative importance of the various criteria may be different. For a particular application, it may be desirable to redesign the metrics to take this into account, for instance reducing the computation component and focusing more on the communication. The following comparison is more a qualitative comparison. A more thorough comparison was performed, including with the Hierarchic Scheme, in Sections 4.4 and 4.6.

3.10.3 Performance Metric

The performance of an algorithm is computed from the steady-state accuracy reached after computation. However, this comparison is a qualitative first comparison. The various algorithms were implemented on a simple example, to get insights, with three vehicles plus the Master (the number of vehicles was chosen arbitrarily). The accuracies obtained were between 10^{-3} m and 10^{-2} m; these values were normalized, to get values between 0 and 1 using

$$\texttt{Performance} = \log\left(\frac{\texttt{accuracy}}{10^{-3}}\right)$$

The values for the performances ξ obtained for the different algorithms were

- Centralized (Section 3.3) $\xi = \log\left(\frac{2.5 \cdot 10^{-3}}{10^{-3}}\right) = 0.40$;
- Tightly Synchronized (Section 3.4) $\xi = \log\left(\frac{3 \cdot 10^{-3}}{10^{-3}}\right) = 0.48$;
- Low-Level Synchronization (Section 3.5) $\xi = \log\left(\frac{4 \cdot 10^{-3}}{10^{-3}}\right) = 0.60$;
- Fully Fleet Asynchronous (Section 3.6) $\xi = \log\left(\frac{10^{-2}}{10^{-3}}\right) = 1$;
- Constant Time-Lag (Section 3.7) $\xi = \log\left(\frac{9 \cdot 10^{-3}}{10^{-3}}\right) = 0.95$;
- Partially Synchronized (Section 3.8) $\xi_{\text{Fully Fleet Asynchronized}} \succ \xi \succ \xi_{\text{Tightly Synchronized}}$;

3.10.4 Penalty Metric

The Penalty metric evaluates the draw-backs of the various approaches. More precisely, it takes into account at the same time the computational load, the communication burden, and the synchronization degree. This is an extension of the basic decentralization penalty developed in Ref. [5].

First consider the synchronization degree, and define it by the number of times a satellite waits for information from others during one cycle of the estimation loop (a measurement update and time propagation). This value is normalized to the range $[0, 1]$, where a value of 0 is associated to an algorithm that does not require satellites wait for others. A value within $[0, 1/3]$ is used when some satellites wait for a constant

time-lag (not for a given other satellite). A value between 1/3 and 2/3 is given when the number of times a satellite waits for others is of order the size of the fleet n. A value larger than 2/3 is used when the algorithm waits for a number of satellites that is on the order of n^2. Then

- Centralized $\sigma = 1/3$;
- Tightly Synchronized $\sigma = 3/4$;
- Low-Level Synchronization $\sigma = 2/3$;
- Fully Fleet Asynchronous $\sigma = 0$;
- Constant Time-Lag $\sigma = 0.1$;
- Partially Synchronized $1/3 \succ \sigma \succ 0$;

To evaluate the computational load, the analysis in [10] was used, and a similar one was performed for the Centralized Algorithm. For each algorithm, the expression of the complexity as a function of the sizes of the different matrices implied in the computation was derived. Then, only the order of the greatest term for n large was considered, to get an idea of how complex the computation becomes for large fleets.

For the Schmidt-Kalman Filter, the complexity computed in [10] is a polynomial function of the different sizes of the matrices used. The largest term in this function is of order $O(n^3)$. For the Centralized Algorithm, the highest order term comes from the inversion of a term of same size as R in the computation of the Kalman gain. This matrix, in this case, is of size the number of measurements across the fleet, that is $O(n^2)$; then, to make the inversion, the complexity becomes $O(n^6)$. Finally, the other algorithms use the same scheme as the SKF and are of same complexities. Then

- n^6 for the Centralized Algorithm;
- n^3 for the other algorithms.

Taking the power as a way to evaluate this complexity, and normalizing by the centralized calculation yields the following computational loads

- Centralized $\pi = 1$;
- Tightly Synchronized $\pi = 0.5$;

- Low-Level Synchronization $\pi = 0.5$;
- Fully Fleet Asynchronous $\pi = 0.5$;
- Constant Time-Lag $\pi = 0.5$;
- Partially Synchronized $\pi = 0.5$;

The communication burden is evaluated through the maximum size of the data needed by the satellites to compute. In all the algorithms, each satellite needs to know the state and covariance of the other satellites, except the Centralized one, which needs all the observations from the spacecrafts (data size $n(n-1)/2$, the total number of measurements performed among the fleet). Also, the Tightly Synchronized Algorithm performs I iterations. Thus, in the Centralized Algorithm, the Master needs $n(n-1)/2$ measurements; in the Tightly Synchronized Algorithm, every satellite needs $I(n-1)$ state estimates and related covariance; in all other algorithms, each satellite needs $(n-1)$ state estimates and covariances; in order to capture an idea of those differences (I and n can vary: a universal metric cannot be developed),the value of 1 was attributed to the most demanding case (Centralized Algorithm), 0 for the least demanding, and 0.25 for the Tightly Synchronized Algorithm (whose maximum data size required –$I(n-1)$– is closer from $(n-1)$ than from $n(n-1)/2$, for n large). This leads to the following communication penalties

- Centralized $\mu = 1$;
- Tightly Synchronized $\mu = 0.25$;
- Low-Level Synchronization $\mu = 0$;
- Fully Fleet Asynchronous $\mu = 0$;
- Constant Time-Lag $\mu = 0$;
- Partially Synchronized $\mu = 0$;

Finally, to get a Penalty metric, all those values were added and normalize by 3

$$\texttt{Penalty} \equiv \rho = \frac{\sigma + \pi + \mu}{3}$$

Combining the above results gives the following values

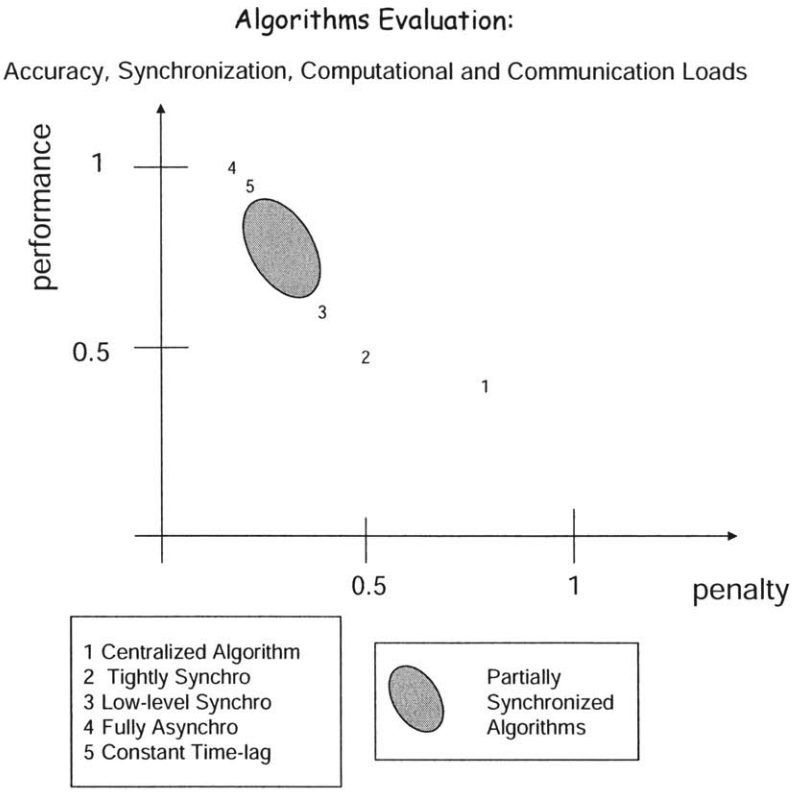

Figure 3-3: Evaluation of the Algorithms

- Centralized $\rho = 0.78$;
- Tightly Synchronized $\rho = 0.5$;
- Low-Level Synchronization $\rho = 0.39$;
- Fully Fleet Asynchronous $\rho = 0.17$;
- Constant Time-Lag $\rho = 0.2$;
- Partially Synchronized $\rho_{\text{Tightly Synchronized}} \succ \rho \succ \rho_{\text{Fully Fleet Asynchronized}}$;

The performance versus penalty graph in Fig. 3-3 shows a very interesting trend. The predictions lie on a Pareto-optimal curve: as the penalty decreases, the performance degrades. This suggests that there is a clear trade-off to be made in the selection of what level of accuracy is desired and the effort required to implement it.

3.11 Conclusion for the Decentralized Scheme

In this chapter, several algorithms based on a Decentralized Scheme were qualitatively compared. The major highlights of this comparison are

- The Centralized Algorithm, designed to provide optimality by gathering all information at the same place, relies on a heavy computation, which becomes unaffordable for large fleet sizes.
- The Iterative Schmidt-Kalman Filter Algorithm addresses this computational issue, but requires a high level of synchronization, and a lots of communication between the spacecraft.
- The Low-Level Synchronization Algorithm, the Partially Fleet Synchronized Algorithm, and the Fully Asynchronous Algorithm present various possibility to address the synchronization concern. The cost of this is a decreased accuracy in these schemes.

Although these comparisons rely on rather loose metrics, these qualitative comparison emphasize the possibility to address the computational issue of the Centralized Algorithm. However, this is usually done by requiring more synchronization and/or communication. In this context, it would be desirable to design algorithms decreasing these requirements, while maintaining a good accuracy in the estimate.

Chapter 4 presents further detailed comparisons with the Hierarchic Scheme. In summary, the algorithms chosen from this chapter are:

- The Centralized Algorithm;
- The Iterative Schmidt Kalman Filter;
- Another version of the Schmidt Kalman Filter, called in Chapter 4 the "Decentralized Algorithm" will be compared as well;

Chapter 4

Hierarchic Clustering and Architectures Comparison

4.1 Introduction

In the previous chapter, various algorithms were qualitatively compared. The Decentralized Scheme was shown to be effective in spreading the computational load of the Centralized Algorithm. However, other concerns arose for such Decentralized Algorithms, in particular the increased synchronization constraint and communication burden. The synchronization constraint could be reduced in an asynchronous form of the Decentralized Algorithm, but at the cost of a deteriorated accuracy.

In this context, the Hierarchic Scheme, whose performance was briefly assessed in the previous chapter, appears to be promising. Proposed in [37], such algorithms could indeed efficiently reduce the computational load of the Centralized Algorithm, while maintaining a low synchronization constraint and communication burden, and reasonable accuracy.

The goal of this chapter is to introduce more precisely the Hierarchic Architectures, and to compare in a more thorough way the Centralized Algorithm with some Decentralized and Hierarchic Algorithms. To compare the various algorithms consid-

ered, the criteria of interest, enumerated in Chapter 3, are used to design metrics.

4.2 Hierarchic Clustering

4.2.1 Presentation

Proposed in [37] and [5] for instance, the Hierarchic Scheme is based on the idea of dividing the fleet into several Sub-Clusters, as presented on Fig. 4-1. Each Sub-Cluster has its own master, and the set of all the masters forms the "Super-Cluster". Each Sub-Cluster performs an independent filter to estimate its internal state. Information also flows between the Super-Cluster and the Sub-Clusters to ensure proper estimation of the positions of the satellites with respect to the rest of the fleet. Finally, cross-cluster measurements can also be allowed. These cross-cluster rangings can be used to further improve the state estimate. If they are not allowed, each cluster must rely on its internal measurements only. With them, the state estimate of other clusters can be used to correct this estimate.

Various Hierarchic Algorithms are possible, depending on the algorithm (Centralized, Decentralized) used in the Super-Cluster and in the Sub-Clusters. The various algorithms are named after the algorithms used for both the Super-Cluster and the Sub-Clusters. For instance, "Centralized-Centralized" designates the Hierarchic Algorithm were both the Super-Cluster and the Sub-Clusters perform the Centralized Algorithm.

4.2.2 Definition of the Algorithms

The basic principle of a Hierarchic Algorithm is to perform the algorithms described in Chapter 3 in the Super-Cluster and in the Sub-Clusters. Each Sub-Cluster and the Super-Cluster thus implements either the Centralized Algorithm or one of the Decentralized Algorithms, as described in Chapter 3. Depending on the size of the

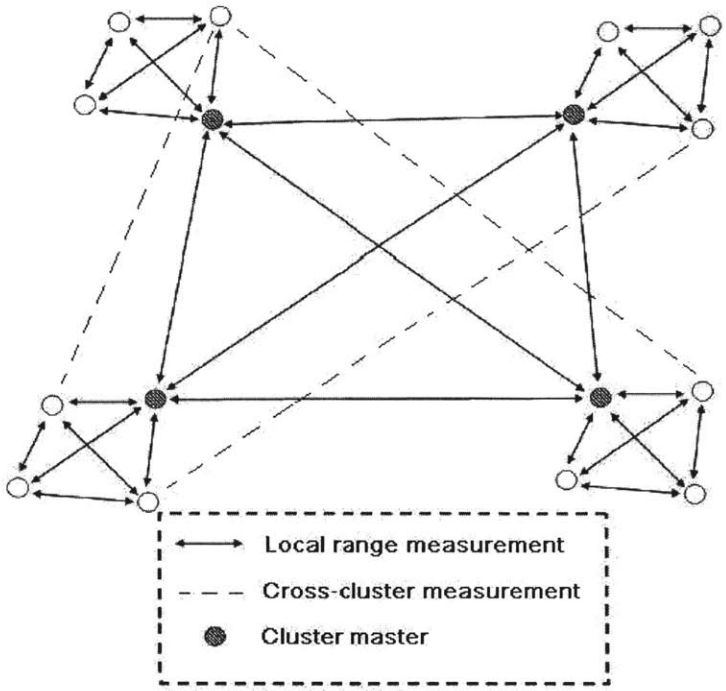

Figure 4-1: Hierarchic clustering scheme, including cross-cluster range measurements. Illustration taken from [4]

Super-Cluster and Sub-Clusters, and on the accuracy, computational complexity, synchronization and communication requirement, one of these options may fit better than the others. On top of these, cross-cluster communication provides every satellite an estimate of its state with respect to the Master. Finally, cross-cluster ranging can be allowed as explained.

Algorithm for the Sub-Clusters

In each Sub-Cluster, the following operations take place in a loop:

- Measurements. Every satellite performs the range-to-station and range-to-satellites measurements.
- Update.
  - When the algorithm chosen for the Sub-Clusters is the Centralized Algorithm, every satellite sends its measurements to the Sub-Cluster's master. The master performs the Time Update and Measurement Update Steps, and sends the estimates back to every satellite.

- When the algorithm for the Sub-Cluster is one of the Decentralized Algorithms, each satellite performs its parts of the computations, as described in Chapter 3. Then, each satellite sends its estimate to the other satellites (or to the next spacecraft for the Iterative Decentralized Algorithm).

Algorithm for the Super-Cluster

Once each Sub-Cluster has computed these operations, the Super-Cluster performs, in a loop:

- Measurements. Every satellite performs the range-to-station and range-to-satellites measurements.
- Update.
  - When the algorithm chosen for the Super-Cluster is the Centralized Algorithm, every satellite sends its measurements to the Master. The Master performs the Time Update and Measurement Update Steps, and sends the estimates back to every satellite.
  - When the algorithm for the Super-Cluster is one of the Decentralized Algorithms, each satellite performs its parts of the computations, as described in Chapter 3. Then, each satellite sends its estimate to the other satellites (or to the next spacecraft for the Iterative Decentralized Algorithm).

Communications from Super-Cluster to Sub-Clusters

After these loops are performed, information needs to be sent from the Super-Cluster to the Sub-Clusters, for each satellite to know its state with respect to the Master. Thus, each master sends to the satellites of the corresponding Sub-Cluster the estimate of its position with respect to the Master. This additional step requires extra-communication as compared with the Centralized and Decentralized Algorithms, as will be emphasized later in this chapter.

4.2.3 Various Synchronization Scenarios

Depending on the synchronization between the Super-Cluster and the Sub-Clusters, several different Hierarchic Schemes can be considered. Depending on the scenario

considered, it can make a difference in terms of performances. This synchronization depends mainly on the existence of cross-cluster measurements. When these measurements are allowed or not, the synchronization requirements differ. More precisely:

- If the cross-clusters range measurements are not allowed, then each satellite is interested in its state relative to its Sub-Cluster, and the Super-Cluster state (in order to know its state relative to the Master).
- If the cross-clusters range measurements are allowed, then every satellite, to range off a satellite from another Sub-Cluster, needs to know its state in its Sub-Cluster, the state of its master in the Super-Cluster, the state of the other Sub-Cluster's master in the Super-Cluster, and the state of the other satellite in its Sub-Cluster, so as to be able to use the range measurement between them. Indeed, to use such a measurement, a satellite needs to know its relative position with respect to the other satellite. For this to work and give a significant improvement in the accuracy, frequent updates on all these estimates are necessary, so that estimate covariances are low. This means that the different algorithms (in Super-Cluster and Sub-Clusters) have to be more tightly synchronized.

So, the main argument in an algorithm definition is the necessary tight synchronization of the algorithms for Super- and Sub-Clusters when the cross-clusters rangings are allowed. Now, when such rangings are not allowed, a certain degree of synchronization would be also appreciable to have a good accuracy in the estimates (since the position of any satellites with respect to the Master is known through the estimate in its Sub-Cluster and the estimate in the Super-Cluster).

Algorithms Nomenclature

In the following nomenclature, an iteration represents for the algorithm considered the set of operations to be performed before moving to the next time step. The different algorithms will be described as follows (p is the number of Sub-Clusters):

- The algorithm corresponding to a Sub-Cluster (either Centralized or Decentralized Algorithm) will be represented by the initials of the Sub-Cluster: for instance, SC for the Super-Cluster, and C_i for the ith Sub-Cluster. This represents one full loop of this algorithm given a set of measurements, including the iterations that the algorithm may need to perform (Iterative Decentralized Algorithm).
- The subscript 1 to this initial refers to one single iteration of the considered algorithm. For instance: SC_1 or C_{2_1}.
- To distinguish between subsequent and parallel algorithms, we use comas and "and": two algorithms with a coma between them are succeeding each other, and two algorithms with "and" are running in parallel.
- Finally, only one loop is described: the full algorithm is obtained by repeating the scheme over time.

For instance, SC, C_1 *and* $C_2 \ldots$ *and* C_p means that the Super-Cluster runs its algorithm fully, iterations included. Then, upon completion, all the Sub-Clusters' algorithms run in parallel, with full iterations. And this keeps happening again. SC_1 *and* C_{1_1}*and* $C_{2_1} \ldots$ *and* C_{p_1} means that all algorithms run in parallel for one iteration. Then, when completed, they all start over a new iteration, and so on.

Algorithms Classification

The algorithms are classified in terms of synchronization between Super-Cluster and Sub-Clusters. Note first that, when a Sub-Cluster is computing, the other Sub-Clusters may be computing at the same time. The only problem is between Super-Cluster and each Sub-Cluster, since they share one spacecraft (the master of the considered Sub-Cluster).

Then, the following Hierarchic Algorithms can be considered:

- The Super-Cluster and the Sub-Clusters compute at the same time.
  The Super-Cluster has priority, Algorithm 1. Then, every time a master has to compute something for both the Super-Cluster and its Sub-Cluster, it computes

for the Super-Cluster first.

The Sub-Clusters have priority, Algorithm 2.

Both algorithms are written SC *and* $C_1 \ldots$ *and* C_p

- The Super-Cluster and the Sub-Clusters compute one after each other.

  They compute the same amount of time.

  + low-level imbrication, Algorithm 3. SC, C_1 *and* $C_2 \ldots$ *and* C_p

  + high-level imbrication, Algorithm 4. SC_1, C_{1_1} *and* $C_{2_1} \ldots$ *and* C_{p_1}

  The Super- and Sub-Clusters do not compute the same amount of time.

  +The Super-Cluster computes for a longer time, either with a low or high-level imbrications, Algorithm 5

  SC, SC, SC, C_1 *and* $C_2 \ldots$ *and* C_p

  +The Sub-Clusters compute for a longer time, either with a low or high-level imbrications, Algorithm 6

  SC, C_1 *and* $C_2 \ldots$ *and* C_n, C_1 *and* $C_2 \ldots$ *and* C_p

Analysis

As said earlier, in terms of accuracy, it is desirable to have the Super-Cluster and in the Sub-Clusters as tightly synchronized as possible, since the estimate of a satellite's state with respect to the Master goes through the estimates in its Sub-Cluster and in the Super-Cluster. It is even more important when cross-clusters measurements are allowed, as argued. On the other hand, having the algorithms more tightly synchronized is demanding, and thus not always desirable. Moreover, the cost of allowing cross-cluster range measurements in terms of synchronization requirement should be further studied. Note simply here that it may be too demanding for the gain in accuracy.

Now, when the Hierarchic Architecture is such that the masters of the Sub-Clusters compute only for the Super-Cluster or for the Sub-Cluster but not for both, it seems easy to have Super-Cluster and Sub-Clusters compute at independent paces ("together", algorithms 1 and 2). It also seems desirable, since we always want to have recent updates, and the only reason why it could be hard is synchronization. Thus,

one has to decide which has priority. It seems that the Super-Cluster, because of its central role, should have the priority, so the algorithm will be implemented with a priority given to the Super-Cluster. If the masters of the Sub-Clusters are used in both the Super-Cluster and in Sub-Clusters, then a synchronization issue occurs, and it is important to distinguish between the different synchronization scenarios. For this thesis, mostly the Hierarchic Centralized-Centralized Algorithm was studied in detail, so that this issue was not crucial.

4.3 Evaluation Metrics

This section presents the evaluation metrics designed to evaluate the performance of the algorithms considered with respect to each of the criteria of interest for this study (see Chapter 3):

- Accuracy
- Computational Complexity
- Synchronization Constraints
- Communication Burden

The goal of this analysis is not to summarize the evaluation of a given algorithm with a single value, but rather to provide a set of insightful metrics which would help the user understanding which algorithm fits best with a given scenario.

The definitions of most metrics use the term "loop". Loop here means the set of operations an algorithm needs to perform before moving to the next time step. For instance, in the Iterative Decentralized Algorithm, a loop will comprise all the iterations over the fleet.

4.3.1 Accuracy Metrics

Problem Statement: the first criterion on the list is the accuracy of the estimate obtained by the algorithm. This accuracy varies in time by the succession of the time

update steps (which increase the uncertainty) and the measurement update steps (which reduce the uncertainty). In some applications, only the accuracy averaged over time is of interest. For other applications, the limiting factor is the worst accuracy reached over time. Thus, two metrics are defined for this criterion.

Accuracy Metrics Definition: the accuracy metrics are based on the covariance matrix, P, which is by definition:

$$P = E\left[\hat{\mathbf{x}}\hat{\mathbf{x}}^T\right]$$

The two metrics defined to quantify the accuracy are:

- "Average Accuracy" is the average of the accuracy over time when steady-state is reached. This accuracy is computed as:

$$AveAcc = \frac{1}{t_{end}} \sum_{t=1}^{t_{end}} \sqrt{\sum_{i=1}^{n} P(i,i)},$$

  this average being computed after each time update and after each measurement update.
- "Worst-Case Accuracy" is the average of the accuracies obtained right before the measurement updates take place. The same formula as for the average accuracy is used, but the average is computed only right before the measurement update steps. This metric takes into account the time needed to compute the measurement update step. It represents the worst error the algorithm can make over time.

Computation of the Accuracy Metrics: to compute the accuracy of a given architecture, we would have to solve the Riccati equations, which are very complicated, involving huge matrices and linearizations of nonlinear functions. This is usually hard to do. Moreover, the covariance that would result from such an analysis would be only the predicted covariance, which may differ from the actual covariance. Thus, the accuracies of the Centralized and Decentralized Architectures were compared through

simulation. These simulations used the same random values, so as to be fair in the comparisons. Several simulations were run for each value, so as to get a statistically meaningful result.

4.3.2 Computational Complexity Metric

Computational Complexity Metric Definition: the computational complexity metrics aims at capturing the load of computations the satellites have to perform for each algorithm. To capture this load, the metric chosen was the average maximum time taken by every loop of the satellites before moving to the next time step. The maximum is taken over the fleet. For instance, for the Centralized Algorithm, the Master has to perform all the computations, so that the metric will be equal to the time taken by the Master to compute. This metric thus captures the maximum complexity of the computations the satellites have to perform before moving to the next time step.

Computation of the Computational Complexity Metric: for each algorithm, the number of elementary operations to be performed can be computed, and the computational complexity metric is proportional to this number. Such an analysis is presented in Section 4.4. In the simulations, the time taken by each satellite to perform one loop was directly measured. The results are presented in Section 4.6.

4.3.3 Synchronization Metric

Synchronization Metric Definition: the synchronization metric represents the average total number of times a satellite has to wait for other spacecrafts to keep on performing its computation task, for every loop. The synchronization criterion aims at capturing the degree to which satellites depend on exterior events to occur before they can start computing. This does not include the effect of a delay in some satellite's task upon another one. The synchronization metric is thus the number of

time a satellite has to wait for another one, per loop.

Computation of the Synchronization Metric: the synchronization metric can be computed through direct analysis (see Section 4.4). It was also measured in the simulations (see Section 4.6).

4.3.4 Communication Metric

Communication Metric Definition: the communication metric captures the amount of information exchanged between satellites. It is defined similarly to the synchronization metrics by the average total amount of information transmitted through the fleet, per loop. An overhead is added to the size of every information sent, corresponding to the overhead accompanying every communication from a satellite to another.

Computation of the Communication Metric: the communication metric is computed through direct analysis in Section 4.4 and by way of simulations in Section 4.6.

4.3.5 Metrics Normalization

All these metrics are normalized, in order to give more straight-forward values. The normalization values are the following:

- Accuracy. The accuracy (both average and worst-case) is normalized by an estimate of the accuracy reachable for a fleet of satellites with given measurements and motion noise covariances.
- Computational Complexity. The computational complexity metric is normalized by an estimate of the computational complexity of the Centralized Algorithm for 20 satellites.
- Communication Load. The communication metric is normalized by the communication needed for the Centralized Algorithm for 20 satellites.
- Synchronization metric. The synchronization metric is normalized by the synchronization for the Centralized Algorithm for 20 satellites.

4.3.6 Metrics Summary

As a summary, the following metrics were defined

- Accuracy Metrics. Two metrics were computed to evaluate the accuracy of a given algorithm, both computed when the algorithm has reached steady-state
  - Worst-Case Accuracy. The worst case accuracy metric consists of the average of the accuracy right before every measurement update step.
  - Average Accuracy. This metric consists of the average accuracy over time.
- Computational Complexity Metric. The computational complexity metrics consists of the average maximum time it takes for every satellite to perform one loop of the algorithm.
- Synchronization Metric. This metric consists in the average total number of time a satellite waits for another to compute, per loop.
- Communication Metric. This metric is the average size of the data transmitted over the fleet, overhead included, per loop.

For each algorithm, these metrics were computed. The set of metric values provides information to compare the algorithms with respect to the different criteria. Then, depending on the mission of interest (computers performance, importance dedicated to the navigation algorithm, desired accuracy, etc), the user can decide which algorithm is best suited to his needs.

4.4 Analytical Studies

This section presents several analytical studies. The goal is to understand how the algorithms could compare with each other in terms of the criteria in 4.3. However, in many cases, since such analysis is usually hard to perform analytically, simplified models were used to obtain insights into the comparison between the algorithms.

4.4.1 Choosing the Number of Clusters

Designing Hierarchic Architectures, as presented earlier in this chapter, one major parameter to chose is the number p of Sub-Clusters into which the fleet is divided. In this section, some analysis are presented to chose this parameter. The number of Sub-Clusters is basically chosen to optimize the accuracy, computational load, synchronization constraint, and communication burden of the Hierarchic Algorithm. In these studies, the algorithms used in the Super-Cluster and in the Sub-Cluster are assumed to be the same. Most arguments presented here are rather loose, since a precise analysis would chose a number of Sub-Clusters for each scenario. Here, only one number was chosen for all scenarios. N is the number of satellites in the fleet.

Optimizing the Accuracy

The accuracy metrics consists of the error between the estimate and the truth. In a Hierarchic Architecture, to know one satellite's position with respect to the Master, this satellite has to know its position with respect to its master, and the position of this master in the Super-Cluster. Since the covariance of the sum of two independent measurements is the sum of the covariances, the predicted covariance for the Hierarchic Algorithm is

$$Cov_H(N) = Cov(p) + Cov(N/p) \tag{4.1}$$

where $Cov_H(N)$ is the covariance of the Hierarchic Architecture for N satellites, and $Cov(x)$ is the accuracy of the considered architecture for the Sub-Clusters and for the Super-Cluster (we assume the same type of algorithms for both, here) for x satellites. Considering p a real-valued parameter, for analysis purposes, and taking the derivative of $Cov_H(N)$, Equation 4.1 gives

$$\frac{\partial Cov_H(N)}{\partial p} = \frac{\partial Cov(p)}{\partial p} - \frac{N}{p^2}\frac{\partial Cov\left(\frac{N}{p}\right)}{\partial p}$$

This expression goes to zero for $p = \sqrt{N}$. To show that $p = \sqrt{N}$ leads to the minimum covariance, the second derivative is computed

$$\frac{\partial^2 Cov_H(p)}{\partial p^2} = \frac{\partial^2 Cov(p)}{\partial p^2} + \frac{N^2}{p^4}\frac{\partial^2 Cov\left(\frac{N}{p}\right)}{\partial p^2} - \frac{2N}{p^3}\frac{\partial Cov\left(\frac{N}{p}\right)}{\partial p}$$

Now, a larger fleet provides more information to estimate the state of the fleet. Thus, $Cov(p)$ decreases with p. There is also a limit accuracy one cannot go beyond, even with a high number of satellites: thus, $Cov(p)$ has a lower bound. As a result, $Cov(p)$ is a monotonically decreasing function with a lower bound, so that $\frac{\partial Cov(p)}{\partial p} < 0$ and $\frac{\partial^2 Cov(p)}{\partial p^2} > 0$. As a result, $\frac{\partial^2 Cov_H(p)}{\partial p^2} > 0$. Finally, $p = \sqrt{N}$ leads to the minimum covariance, so that it is the best choice, under the assumptions made here, in terms of accuracy.

Optimizing the Computational Complexity

The computational complexity metric consists of the (average) maximum time it takes for a satellite to perform its computational task. Thus, the predicted complexity of the Hierarchic Architecture is roughly

- If the satellite computing most is computing for the Super-Cluster and for a Sub-Cluster:

$$Comp_H(N) = Comp(p) + Comp(N/p)$$

- If the satellites computing most is involved either in the Super-Cluster's algorithm or in a Sub-Cluster's algorithm, but not in both of them:

$$Comp_H(N) = \max\left[Comp(p), Comp(N/p)\right]$$

- If the satellite computing most is computing for the Super-Cluster and for a Sub-Cluster: when the number of satellites increases, the computational burden increases as well, for all considered algorithms, so that $\frac{\partial Comp(p)}{\partial p} > 0$. Moreover, the computational complexity is assumed to be a polynomial function of order greater than one, so that $\frac{\partial^2 Comp(p)}{\partial p^2} > 0$. As a result, similar calculations as for

the accuracy criterion lead to:

$$\frac{\partial^2 Comp_H(N)}{\partial p^2} \geq 0$$

- If the satellite computing most is involved either in the Super-Cluster's algorithm or in a Sub-Cluster's algorithm, but not in both of them Again, the computational complexity is increasing with the fleet size. As a result
  - if $p \leq \sqrt{N}$, then $\max\left[Comp(p), Comp(N/p)\right] = Comp(N/p) \geq Comp\left(\sqrt{N}\right)$.
  - if $p \geq \sqrt{N}$, then $\max\left[Comp(p), Comp(N/p)\right] = Comp(p) \leq Comp\left(\sqrt{N}\right)$

  As a result, the optimal number of Sub-Clusters under these assumptions is again $p = \sqrt{N}$.

Optimizing the Synchronization Constraints

The synchronization metric consists of the number of time a satellite waits for another, per loop. Thus, the expected synchronization metric for the Hierarchic Architecture is

$$Syn_H(N) = Syn(p) + pSyn\left(\frac{N}{p}\right)$$

Taking the derivative, one gets

$$\frac{\partial Syn_H(N)}{\partial p} = \frac{\partial Syn(p)}{\partial p} + Syn\left(\frac{N}{p}\right) - \frac{N}{p}\frac{\partial Syn\left(\frac{N}{p}\right)}{\partial p}$$

The value of p for which $\frac{\partial Syn_H(N)}{\partial p} = 0$ is not clear. One can simply note that:

$$Syn_H(1) = Syn_H(N) = Syn(N)$$

and:

$$\frac{\partial Syn_H(N)}{\partial p} = \frac{\partial Syn(N)}{\partial p} - \frac{\partial Syn(1)}{\partial p},$$

which is positive if one assumes Syn to be polynomial of order greater than one. Thus, the value of p for which Syn is minimal is strictly between 1 and N. Moreover,

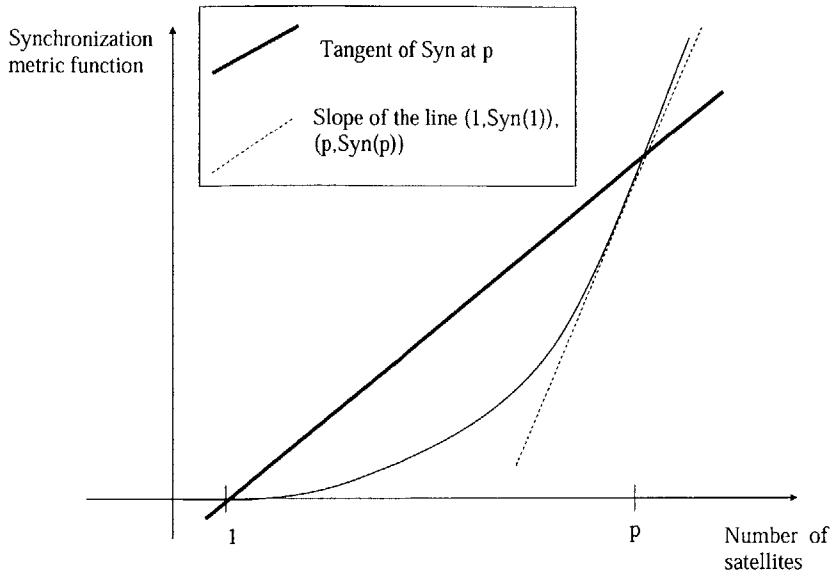

Figure 4-2: Synchronization is assumed to be a polynomial function of order greater than one

computing the derivative of $Syn_H(p)$ for $p = \sqrt{N}$:

$$\frac{\partial Syn_H(p)}{\partial p}\left(\sqrt{N}\right) = (1-\sqrt{N})\left(\frac{\partial Syn(\sqrt{N})}{\partial p} - \frac{Syn(\sqrt{N}) - Syn(1)}{\sqrt{N}-1}\right)$$

(with $Syn(1) = 0$). Now, if Syn is polynomial of order greater than one, then (see Fig. 4-2):

$$\frac{\partial Syn(\sqrt{N})}{\partial p} - \frac{Syn(\sqrt{N}) - Syn(1)}{\sqrt{N}-1} > 0$$

so that:

$$\frac{\partial Syn_H(p)}{\partial p}\left(\sqrt{N}\right) < 0$$

As a result, under these assumptions, the optimal value of p is strictly between $\sqrt{N}$ and N.

Optimizing the Communication Burden

The communication metric consisting of the size of the data transmitted over the

fleet, the predicted communication metric for a Hierarchic Architecture is

$$Comm_H(N) = Comm(p) + pComm(N/p)$$

With again the assumption that the communication metric is a polynomial function of order greater than one, the same analysis as for the synchronization metrics holds: the optimal number of clusters cannot be determined by this method, but it is known to be greater than $\sqrt{N}$.

Conclusion

In terms of accuracy and complexity, the optimal choice was shown to be $p = \sqrt{N}$, with the assumptions made here. In terms of synchronization and communication, the optimal choice is not clear, but is greater than $\sqrt{N}$. For the following studies, it was chosen here to take $p = \sqrt{N}$. Further study could be performed to refine this choice for each algorithm.

4.4.2 Accuracy

Assessing with theoretical tools the accuracy of a given algorithm is particularly challenging because of the complexity of the equations involved. Indeed, solving the discrete steady-state Riccati equations should be needed to compare the various algorithms. In this section, a simplified one-dimensional model is presented, to get insight into the comparison in the accuracy between non-Hierarchic and Hierarchic Algorithms. In the system presented here, the information form was found to lead to easier computations, so that it was used to find the approximate accuracy.

Presentation of the Simplified Model

The model considered is made of a fleet of N one-dimensional non-moving vehicles, defined by their state x (scalar). Fig. 4-3 represents this system. The estimation consists of the state estimation $\hat{x}$ and the related covariance P.

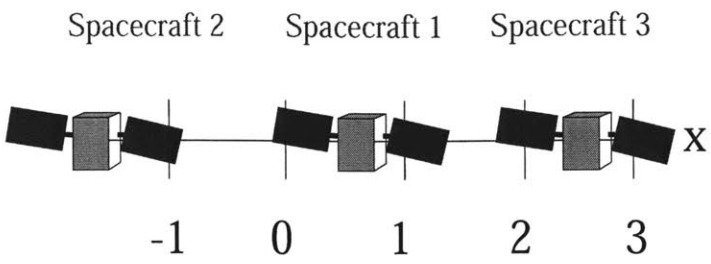

Figure 4-3: One dimensional model representation

The goal of this study is to evaluate the covariance resulting from the computation of an estimate for the vehicles positions. The measurement update step was written in the information form, since it was found easier to handle in this analysis. The motion noise is modeled with a constant covariance, so that the Time Update Step is

$$P^+ = P^- + Q$$

The noise covariance is modeled as $Q = Q_1 I$. The measurements available at each time step are the linear differences between the different states

$$r_{ij} = x_i - x_j + v_{ij}$$

and the linear difference between each satellite and the origin:

$$r_{i0} = x_i + v_i$$

This latter measurement was added to make the system observable. The information (information form of the Kalman Filter, see for instance [28]) available from these measurements is

$$\Gamma = H^T R^{-1} H = \begin{bmatrix} N/R & -1/R & \dots & -1/R \\ -1/R & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1/R \\ -1/R & \dots & -1/R & N/R \end{bmatrix},$$

where R is the covariance of the crosslink range measurements.

Solution of the Model for a Centralized Architecture

Let us first show by recursion that the shape of the covariance matrix is

$$P = \begin{bmatrix} P_1 & P_2 & \dots & P_2 \\ P_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & P_2 \\ P_2 & \dots & P_2 & P_1 \end{bmatrix}$$

(all diagonal elements are equal, and all non-diagonal elements are equal). Let us call S the set of matrix with equal diagonal and non-diagonal elements. Assuming that the initial covariance is diagonal $P_0 \in S$. Then, if at some time step $P \in S$ holds, then at the next time step (information form of the Kalman Filter):

$$Y^+ = Y^- + \Gamma$$

where

$$Y = P^{-1} = \begin{bmatrix} \tilde{P}_1 & \tilde{P}_2 & \dots & \tilde{P}_2 \\ \tilde{P}_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tilde{P}_2 \\ \tilde{P}_2 & \dots & \tilde{P}_2 & \tilde{P}_1 \end{bmatrix}$$

where

$$\begin{aligned} \tilde{P}_1 &= \frac{P_1 + (N-2)P_2}{P_1^2 + (N-2)P_1P_2 - (N-1)P_2^2} \\ \tilde{P}_2 &= \frac{-P_2}{P_1^2 + (N-2)P_1P_2 - (N-1)P_2^2} \end{aligned}$$

Thus, $P \in S$ implies $Y \in S$. Since $\Gamma = H^T R^{-1} H \in S$, one gets $Y^+ \in S$. Finally,

$Y^+ \in S$ implies $P^+ \in S$. It was thus shown that at every step, P has the shape:

$$P = \begin{bmatrix} P_1 & P_2 & \dots & P_2 \\ P_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & P_2 \\ P_2 & \dots & P_2 & P_1 \end{bmatrix}$$

Let us now consider the steady-state covariance matrix. The equation describing the steady-state is given by blending the motion equation $P^+ = P^- + Q$ and the measurement update equation $Y^+ = Y^- + \Gamma$. This gives:

$$\begin{aligned} \frac{P_1 + (N-2)P_2}{f_N(P_1, P_2)} &= \frac{P_1 + Q_1 + (N-2)P_2}{g_N(P_1, P_2, Q)} + \frac{N}{R} \\ \frac{-P_2}{f_N(P_1, P_2)} &= \frac{-P_2}{g_N(P_1, P_2, Q)} - \frac{1}{R} \end{aligned}$$

with

$$\begin{aligned} f_N(P_1, P_2) &= P_1^2 + (N-2)P_1P_2 - (N-1)P_2^2, \\ g_N(P_1, P_2, Q) &= (P_1 + Q_1)^2 + (N-2)(P_1 + Q_1)P_2 - (N-1)P_2^2 \end{aligned}$$

These equations could not be solved analytically. Thus, numerical solutions were found by implementing the simple Kalman Filter corresponding to this system. The vehicle positions were chosen equally distributed on the segment $[-5; 5]$. The measurement covariance R was chosen equal to 0.1 for all the simulations. The steady-state covariance for each vehicle position, P_1, was plotted as a function of the number N of vehicles, for three different scenarios:

- $Q = 0.01 \ll R$
- $Q = 0.1 = R$
- $Q = 1 \gg R$

The resulting curves are presented on Fig. 4-4. These curves show the improvement in the estimate accuracy as the number of vehicles in the fleet increases.

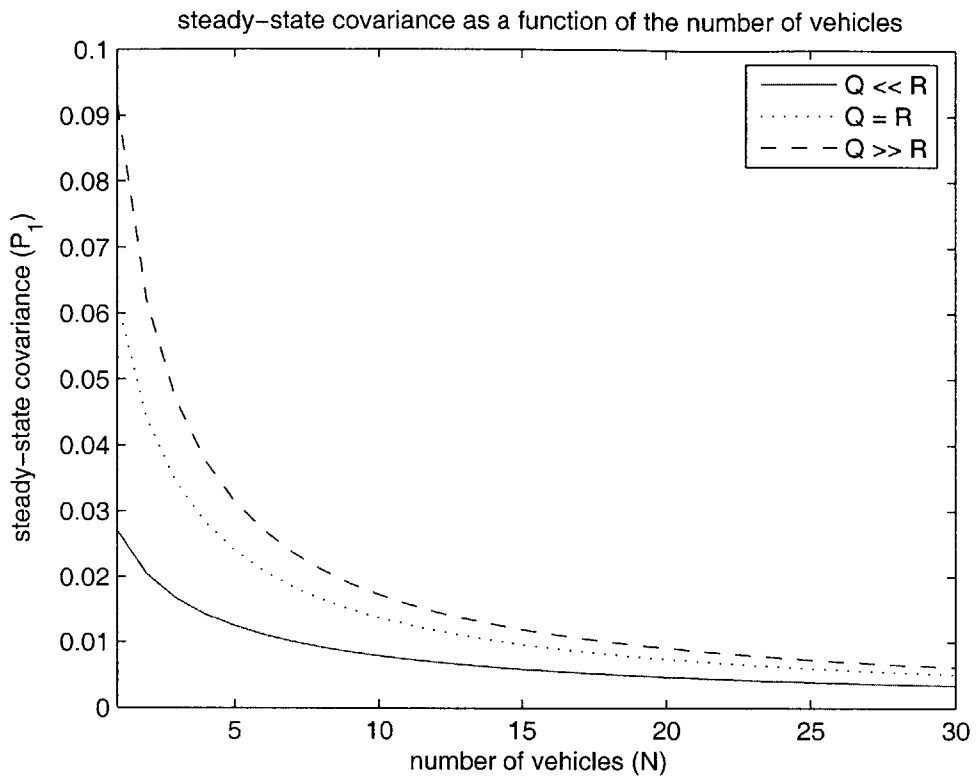

Figure 4-4: Evolution of the steady-state covariance with N for $Q \ll R$

Applying the Numerical Results to Hierarchic Architectures

With these curves, the performance of the Hierarchic Architecture can be estimated. Let us consider, for instance, a fleet of 16 vehicles, with Q being small ($Q = 0.01$). In a Centralized Architecture, from Fig.4-4, the accuracy is around $5.6 \cdot 10^{-3}$. In a Hierarchic Architecture Centralized-Centralized, each Sub-Cluster contains 4 vehicles, so that, still from Fig. 4-4, the accuracy in each Sub-Cluster is $1.4 \cdot 10^{-2}$. The Super-Cluster reaches the same accuracy. Thus, to estimate the position of the vehicles in the Sub-Clusters as defined in the Master's frame, the covariances in the Sub-Cluster and Super-Cluster have to be added. Thus, the covariance becomes, for each vehicle

$$P_1(Hierarchic) = 2.8 \cdot 10^{-2}$$

The difference between a Hierarchic and a Centralized Architecture is fairly high (the

covariance is 5 times greater for the Hierarchic, so that the accuracy is expected to be 2.2 times greater). Note that, as the number of satellites decreases, this difference decreases as well. For instance, for a fleet of 4 vehicles, the covariance in the Hierarchic case would be 2.9 times greater than in the Centralized case (accuracy 1.7 times greater).

When the process noise covariance Q is larger, for instance when $Q = 1$, the steady-state covariance of the estimate decreases much faster with the number of vehicles, as seen on Fig. 4-4. For a fleet of 16 vehicles, the covariance in the Hierarchic case is 7 times greater than in the Centralized case.

Conclusions

This analysis is by no mean a definitive comparison, since it is based on a very simplified model. However, it gives insight into the extent to which accuracy is lost in the Hierarchic Architectures, as compared with the Centralized Architecture. The conclusions from this analysis are the following:

1. As the number of satellites increases, the cost of using a Hierarchic Architecture versus a Centralized one is higher. Note that this is in contrast to the computational complexity: for which, as the number of satellites increases, it is better to use a Hierarchic Architecture.

2. As the process noise decreases, the cost of using a Hierarchic Architecture in terms of accuracy decreases. Indeed, when the process noise is low, the improvement in the accuracy for the state estimate as the number of vehicle increases is lower (compare the curves on Fig. 4-4).

The Hierarchic Architecture is less accurate than the Centralized Algorithm mainly because some range measurements are not used in the former. Note that this analysis was performed with 2-D vehicles. In 2-D, the fleet having fewer degrees of freedom, the improvement in accuracy as the number of measurements increases is less pronounced. Thus, this effect is excepted to be more important in 3-D.

4.4.3 Computational Complexity

This section presents an analysis of the compared computational complexities of the different algorithms, based on counting the elementary operations to be performed. A more general but similar analysis for the different types of Kalman Filters can be found at [10]. The state vector considered here is of size $s = 13$. It contains position (3 states), velocities (3), quaternions for attitude (4), and angular velocities (3)

Centralized Algorithm

In the Centralized Algorithm, an Extended Kalman Filter is implemented for the whole fleet. The complexity can be computed as follows (following the methodology in [10]):

1. Time Update Step
   - State Update

     $$\hat{x}(k|k-1) = A\hat{x}(k-1|k-1),$$

     where $\hat{x}$ contains the states of all the satellites in the fleet, A is the propagation matrix. The complexity of this step is: $(sN)^2$ to compute $A\hat{x}(k-1|k-1)$.
   - Covariance Update

     $$P(k|k-1) = AP(k-1|k-1)A^T + Q,$$

     where P is the covariance matrix, and Q is the noise covariance. The complexity of this step is: $(sN)^3$ for $AP(k-1|k-1)$, $(sN)^3$ for $(AP(k-1|k-1))\,A^T$, and $(sN)^2$ to add Q.
   - The total complexity for the Time Update Step is $O(2(sN)^3)$.
2. Measurement Update Step
   - Kalman Gain

     $$K = P(k|k-1)H^T\left(HP(k|k-1)H^T + R\right)^{-1}$$

The complexity of the computation of the Kalman gain is $(sN)^2N(2N+4)$ to compute $(P(k|k-1)H^T)$, $(sN)\left(N(2N+4)\right)^2$ for $H\left(P(k|k-1)H^T\right)$, $(N(2N+4))^2$ to add R, $(N(2N+4))^3$ to inverse, $(sN)\left(N(2N+4)\right)^2$ to multiply by $P(k|k-1)H^T$.

- State Update

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K\left(z_k - \hat{z}_k\right),$$

where K is the Kalman gain, z_k is the measurement vector at time k, and $\hat{z}_k$ is the predicted measurement at the same time. The complexity of this step is $N(2N+4)$ to compute $\hat{z}_k$, $N(2N+4)$ to perform$(z_k - \hat{z}_k)$, $(sN)N(2N+4)$ to multiply by K, $N(2N+4)$ to add $\hat{x}(k|k-1)$.

- Covariance Update

$$P(k|k-1) = (I - KH)\,P(k|k-1).$$

The complexity of this step is $(sN)^2N(2N+4)$ to compute KH, (sN) to subtract it to I, and $(sN)^3$ to multiply it to $P(k|k-1)$.

- The total complexity for the Measurement Update Step is $O(8N^6+8sN^5+4s^2N^4+(sN)^3)$.

3. The total complexity for the Centralized Algorithm is $O(8N^6+8sN^5+4s^2N^4+3(sN)^3)$.

Decentralized Algorithms

In a Decentralized Algorithm, each satellite implements a Schmidt-Kalman Filter [37, 10].

1. Time Update Step.

   - State Update (each satellite performs the time update corresponding to the state estimates of all the satellites)

$$\hat{x}_i(k|k-1) = A\hat{x}_i(k-1|k-1) + a_i(k),$$

for every satellite i in the fleet. The complexity of this step is s^2N for $A\hat{x}_i(k-1|k-1)$, sN to add the acceleration.

- Covariance Update

$$P_i(k|k-1) = AP_i(k-1|k-1)A^T + Q_i,$$

The complexity here is s^3N for $AP_i(k-1|k-1)$, s^3N to multiply by A^T, s^2N to add Q.

- The total complexity for the Time Update Step is $O(2s^3N)$.

2. Measurement Update Step

- Kalman Gain

$$\alpha = HP_{xx}H^T + HP_{xy}J^T + JP_{yx}H^T + JP_{yy}J^T + R$$

The complexity to compute α is $s^2(2N+4)$ for HP_{xx}, $s(2N+4)^2$ to multiply by H^T, $s^2(2N+4)(N-1)$ for HP_{xy}, $s(2N+4)^2(N-1)$ to multiply by J^T, $s^2(N-1)(2N+4)$ for JP_{yx}, $s^2(2N+4)^2$ to multiply by H^T, $s^2(N-1)^2(2N+4)$ for JP_{yy}, $s(N-1)(2N+4)^2$ to multiply by J^T, $(2N+4)^2$ to add R.

$$K = \left(P_{xx}H^T + P_{xy}J^T\right)\alpha^{-1}$$

The complexity to compute K is ($P_{xx}H^T$ and $P_{xy}J^T$ were already computed for α) $s(2N+4)$ to add $P_{xx}H^T$ and $P_{xy}J^T$, $(2N+4)^3$ to inverse α, $s(2N+4)^2$ to multiply them together.

- State Update

$$\hat{x}^+ = \hat{x}^- + K\left(z - \hat{z}\right)$$

The complexity for the state update is $(2N+4)$ for $z-\hat{z}$, $s(2N+4)$ to multiply by K, s to add $\hat{x}^-$.

- Covariance Update

$$P_{xx} = \left(I - KH\right)P_{xx} - KJP_{yx}$$

The complexity for this update is $s^2(2N+4)$ for KH, s^2 to substract it to I, s^3 to multiply by P_{xx}, $s^2(2N+4)(N-1)$ for KJ, $s^3(N-1)$ to multiply by P_{yx}, s^2 to add the two terms.

$$P_{xy} = (I - KH)\, P_{xy} - KJP_{yy}$$

The complexity here is ($(I-KH)$ has already been computed) $s^3(N-1)$ to multiply by P_{xy}, $s^3(N-1)^2$ for $(KJ)P_{yy}$, and finally $s^2(N-1)$ to subtract the two terms. $P_{yx} = P_{xy}^T$, which necessitates $s^2(N-1)$ operations.

- The total complexity for the measurement update step is $O(2s^2N^3+s^3N^2)$

3. The total complexity for the Decentralized Algorithms is $O(2s^2N^3 + s^3N^2)$

Hierarchic Algorithm: Centralized-Centralized

In the first Hierarchic Algorithm studied, Centralized - Centralized, the Centralized Algorithm is implemented for both the Super-Cluster and the Sub-Clusters. Let p denote the number of Sub-Clusters.

From the previous studies, the complexity can be computed. The Super-Cluster contains p satellites and uses the Centralized Algorithm, so the complexity here is $8p^6 + 8sp^5 + 4s^2p^4 + 3s^3p^3$. Now, each Sub-Cluster will perform the Centralized Algorithm. The Master of the fleet will perform both the algorithm for the Super-Cluster and for the Sub-Cluster to which it belongs, so the complexity is $3s^3N^2 + 4s^2N^{5/2} + 3s^3N^{3/2} + 8sN^3 + 4s^2N^2 + 8N^{7/2} + 8sN^{5/2}$, when $p = \sqrt{N}$.

Note that the cost of computing H or H and J was not directly considered here. In the Centralized Algorithm, the measurement matrix is computed as one single large matrix. In the Decentralized and Hierarchic Algorithms, smaller measurements matrices are computed by various algorithms.

4.4.4 Synchronization Constraint

This section presents the analysis corresponding to the synchronization metrics, which consists of the number of times a satellite waits for another spacecraft to keep on performing its task. By "waiting for" it is meant here that the satellite cannot keep performing computations without receiving the desired piece of information from another satellite.

Centralized Algorithm

In the Centralized Algorithm, for every loop

- The Master must wait for every satellite to send the measurements. Thus, every loop, the Master waits for $N-1$ satellites.
- Every satellite must wait for the Master to send the estimates and covariances. So that $N-1$ satellites wait.

As a result, the synchronization metric for the Centralized Algorithm (before normalization) is $2(N-1)$.

Synchronized Decentralized Algorithms

For the Synchronized Decentralized Algorithm, for every loop, every satellite waits for every other to receive their estimates and covariance. Thus, every loop $N(N-1)$ satellites wait.

Iterative Decentralized Algorithms

In the Iterative Decentralized Algorithm, every satellite waits for every other, at every iteration. The total number of satellites waiting for other is thus $I \times N(N-1)$ per loop, I being the number of iterations.

Hierarchic Centralized-Centralized Algorithm

In the Hierarchic Centralized-Centralized Algorithm, the synchronization metric can be computed by (for a fleet with p Sub-Clusters):

- In each Sub-Cluster, the Master waits for all satellites to send their measurements, $\frac{N}{p} - 1$. Every satellite waits to receive its estimate, $\frac{N}{p} - 1$.
- In the Super-Cluster, the Master waits to receive the measurements of the masters, $p - 1$, and their estimate, $p - 1$. Then, each master waits to receive its estimate, $p - 1$.

Thus, the total number of satellites waiting for each loop is $p(2\frac{N}{p} - 2) + 3(p - 1) = 2N + p - 3$. In particular, this confirms the study performed in Section 4.4.1 to optimize the number of Sub-Clusters: again, one sees that the optimal choice here is to define p as low as possible.

4.4.5 Communication Burden

In this section, a similar study is performed to assess the communication metric for the algorithms studied. The communication load consists of the size of the data transmitted throughout the fleet per loop with a small overhead being added to each communication. This overhead being dependant on the number of satellites in the fleet, it was chosen arbitrarily equal to this number, N. The size of the state of each satellite is again written s ($s = 13$).

Centralized Algorithm

In the Centralized Algorithm, the amount of data communicated is:

- At each time step, every satellite communicates the measurements available at its place. The size of a measurement vector for one satellite is $2N + 4$[1]. Thus, the communication load for this step is $(N-1)(2N + 4 + N) = (N-1)(3N+4)$.
- After computing the estimate, the Master redistributes this estimate to every satellite. So it must send over the communication channel the state estimate $\hat{x}$, of size s, and the related covariance P, of size s^2. With the overhead, the communication load here is $(N - 1)(N + s + s^2)$.

[1]Every satellite measures its range and bearing with respect to each other spacecraft ($2N - 2$ measurements), its absolute attitude (in quaternion, 4 measurements), and two ranges-to-beacons

For every loop, these two sets of exchange take place, so that the total amount of information transmitted per loop is, with the overhead $(N-1)(4N+4+s^2+s) = O(s^2N+4N^2)$.

Synchronized Decentralized Algorithm

In the Synchronized Decentralized Algorithm, each satellite, for every loop sends to every other satellite its estimate and the related covariance. Thus, with the overhead, the communication load is $N(N-1)(N+s^2+s) = O(s^2N^2+N^3)$.

Iterative Decentralized Algorithm

In the Iterative Decentralized Algorithm, the same data as in the Synchronized Decentralized Algorithm is transmitted, for every iteration. Thus, the communication load is $I \times N(N-1)(N+s^2+s) = O(I(s^2N^2+N^3))$.

Hierarchic Centralized-Centralized Algorithm

In a Centralized-Centralized Hierarchic Architecture, the amount of data communicated is

- In each Sub-Cluster, the measurements are sent to the master $\left(\frac{N}{p}-1\right)(2\frac{N}{p}+4+N)$. Then each satellite receives the estimate from the master, $\left(\frac{N}{p}-1\right)(s+s^2+N)$
- In the Super-Cluster, each master sends its estimate to the Master, $(p-1)(s+s^2+N)$, then they send their measurement vector, $(p-1)(2p+4+N)$.
- Finally, every satellite receives its estimate from the Master, $(N-1)(s^2+s+N)$

As a result, the total communication load for the Centralized-Centralized Algorithm is $s^2N+N^2+4N^{3/2}+2s^2\sqrt{N}+2N$.

Comparison Summary

Fig. 4-5, 4-6 and 4-7 summarize the previous analytical studies.

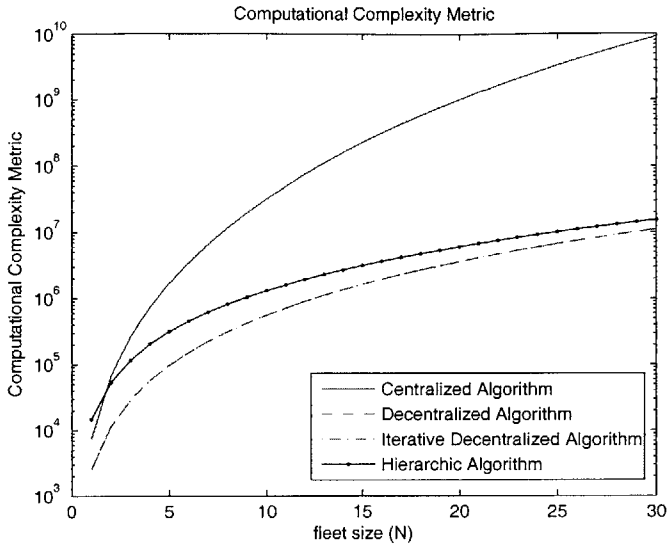

Figure 4-5: Computational Complexity Comparison (logarithmic scale)

4.5 The Spheres Testbed

4.5.1 Introduction

The analysis presented so far corresponds to algorithms for satellites flying in space. In order to test these algorithms, it was decided to prepare experiments with the SPHERES project [41, 42, 43]. As a result, the algorithms implemented for this thesis were inspired by the SPHERES set-up.

4.5.2 The SPHERES program

Many missions of interest for the future of space engineering involve fleets of satellites flying in formation. For instance [44], spacecrafts used as interferometers which could replace a telescope far bigger; Sub-Clusters of satellites performing missions in a more redundant, and cheaper way; tethered spacecraft interferometer, etc.

In this context, MIT has developed a testbed, made of small independent satellites. Each of them is capable of measuring its distance to each other, and to wall beacons.

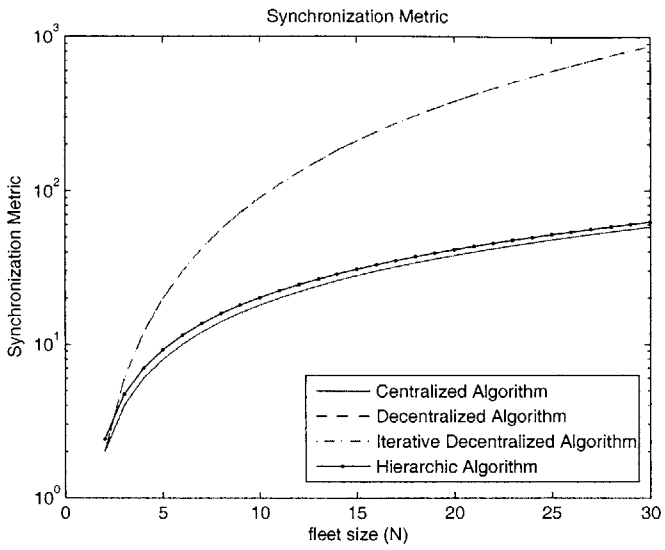

Figure 4-6: Synchronization Comparison (logarithmic scale)

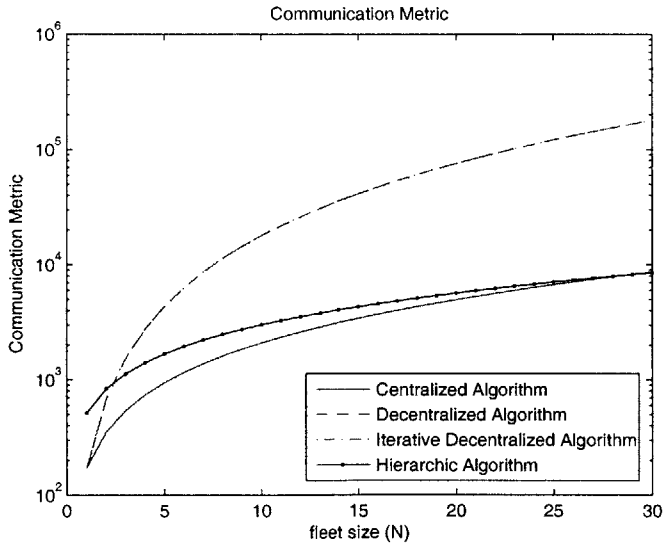

Figure 4-7: Communication Comparison (logarithmic scale)

The range-to-beacons measurements provide enough information for each spacecraft to be able to locate itself. The inter-satellite range measurements are used to improve this estimate and for the docking phases, where two or more satellites are engaged in very precise manoeuver next to each other.

For the purpose of this study, it was desired to put emphasis on the relative measurement infrastructure, using as few absolute measurements as possible, so that any single satellite cannot locate itself based on these absolute measurements only. It was thus decided to use two range-to-beacon measurements, and the crosslink inter-satellite range measurements. This is the set-up simulated in the simulations described in Section 4.6.

4.6 Numerical Results

4.6.1 Simulations Framework

As explained in the previous section, the simulations performed used the SPHERES framework. The SPHERES move in a room of size $1m^3$. No force was applied to the satellites, so that they moved without acceleration. Each satellite uses satellite-to-satellite range measurements and satellite-to-beacon measurements with respect to two wall beacons of known absolute locations, located on the walls of the room. The Decentralized Algorithms showed some instability in computing the global state of the fleet (location of the center of gravity of the fleet). This instability was resolved here by adding a second range-to-beacon measurement, although other methods to address this issue could be studied.

4.6.2 Simulations Road Map

The goal of the simulations is to compare the different algorithms in terms of accuracy, computational load, communication burden, synchronization constraints. Vari-

ous graphs present the performances of the architectures with respect to these criteria.

Methodology

The same code structure was used for all architectures, so as to make the comparison relevant. Each simulation was repeated with a hundred different random seeds so as to get statistical data for each evaluation metric. This provided reliable trends about the algorithm performances. The simulations were performed with an "abstract" version of the SPHERES testbed: N satellites were simulated in 3-D. They could measure their distances to two beacons of known absolute locations. They had access to their absolute attitude. Finally, they measured their distances and elevations with respect to every other satellite in the fleet. The measurement covariance was set to 10^{-4} and the process noise covariance to 10^{-6}. As explained in Section 4.6.4, the computational complexity of the different algorithms introduces different delays in the algorithms. When the algorithm computes in a loop, this delay impacts the real-time accuracy of the estimate. This impact depends on the process noise covariance. Thus, further studies could analyze how the following results scale when the process noise increases. Different fleet size were used, so as to understand how the comparison evolve as this number changes. The fleet sizes used were: 4, 8, 16, 24. The algorithms compared were the:

- Centralized Algorithm;
- Decentralized Tightly Synchronized Algorithm;
- Decentralized Iterative Algorithm (for various number of iterations);
- Hierarchic Centralized-Centralized Algorithm.

For each fleet size, these algorithms are run for various different random seeds. The results presented are the average of these simulations.

Two different scenarios were considered for these comparisons

- In the "varying time step" scenario, for each algorithm, the time step depends on the speed of the algorithm. Thus, each time a loop is over, without waiting, the computer computes the time update step. This scenario gives an advantage for the fast algorithms.

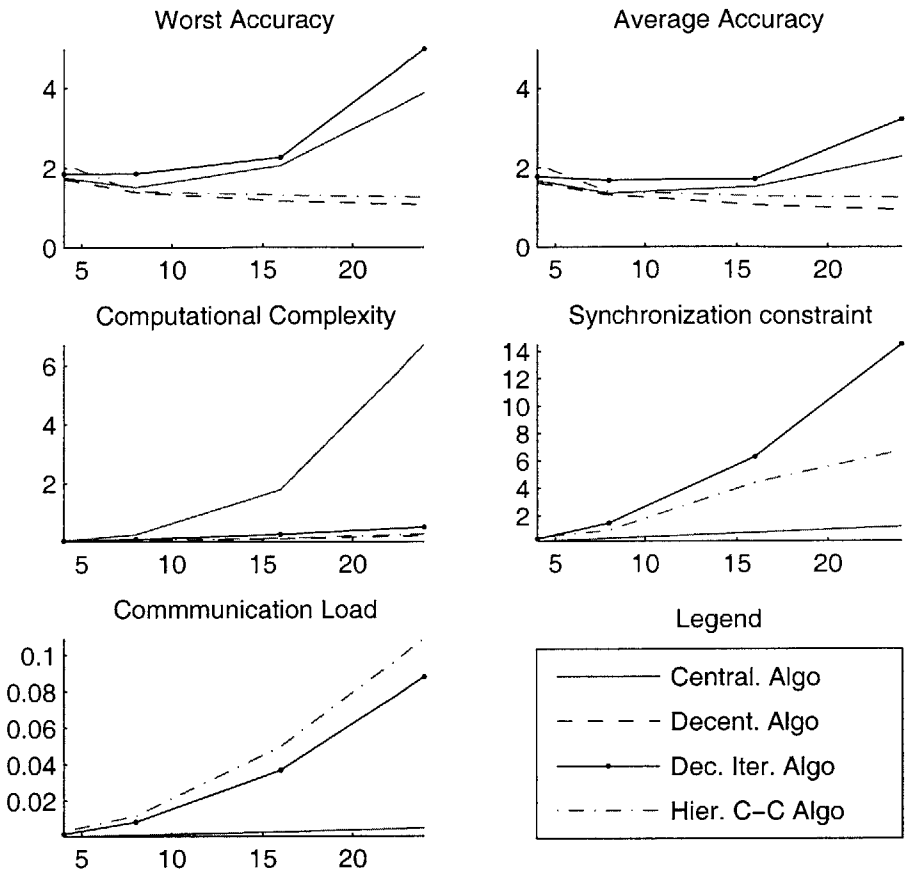

Figure 4-8: Comparison between the algorithms for varying time step scheme

- In the "fixed time step" scenario, the time step is fixed (one second). As a result, even if the computer needs 0.01s to perform the computations corresponding to every time step, it has to wait one second before going to the following time step. This scenario penalizes the fast algorithms, since they have to wait a second before computing the next time step.

4.6.3 Algorithms Comparison for Varying Time Step

The algorithms comparison for this scenario is represented on Fig.4-8.

Comparing the Accuracies

In terms of accuracy, the best algorithm appears to be the Decentralized Tightly Synchronized Algorithm. Indeed, even for low fleet sizes, the Centralized Algorithm has to perform heavier computations, and thus needs longer time, which penalizes its accuracy. For a fleet of four satellites, for instance, it takes 15ms for the Decentralized Algorithm to perform one loop, and 30ms for the Centralized Algorithm. Being only slightly less accurate and twice faster, the Decentralized Algorithm performs better than the Centralized Algorithm.

For large fleets, the computations to be performed in the Centralized Algorithm gets bigger. This computation introduces delays in the algorithm, which eventually impact the real-time accuracy of this algorithm. The Iterative Decentralized Algorithm shows poor performance because of the time needed to go over the whole fleet for each loop. The Decentralized Algorithm performs very well, thanks to the low computational task it has to perform: being slightly suboptimal but much faster than the Centralized, it has much better results. Finally, the Hierarchic Algorithm performs very well too, because the computational task is efficiently spread into the Sub-Clusters.

Comparing the Computational Complexities

For the computational complexity, the algorithm with the heaviest computations is the Centralized Algorithm. The Iterative Decentralized Algorithm shows also a high computational complexity metric because every satellite computes one after the other before coming back to the first one.

Comparing the Communication Loads

The communication load of the Hierarchic Algorithm is high because of the transmissions inside the Sub-Clusters and between the Sub-Clusters and Super-Cluster. The Decentralized Algorithm, and the Iterative Decentralized Algorithm have the same transmissions to perform in one loop so that their metrics are equal. They are high because of the transmission of the states of all satellites to every other satellite. Fi-

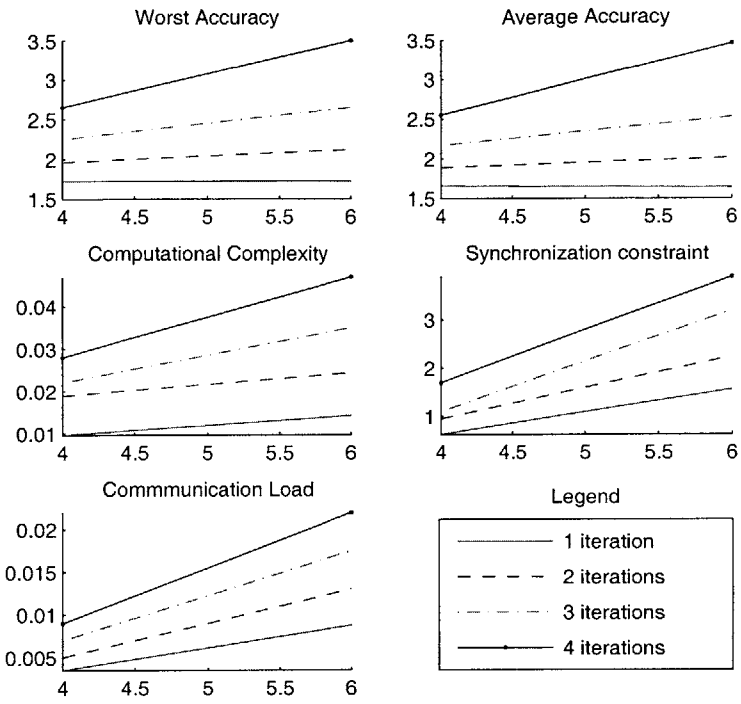

Figure 4-9: Comparison between the Iterative Decentralized Algorithms for varying number of iteration

nally, the Centralized Algorithm shows a low metric value: only the measurements are transmitted to the Master, and the states estimates are transmitted to the satellites.

Comparing the Synchronization Constraints

The synchronization metric results can be explained with similar reason as the communication load. The only difference is for the Hierarchic Algorithm, since it does transmit lots of measurements matrices, so that the communication load is large.

Determining the Optimal Number of Iterations for the Iterative Decentralized Algorithm

Fig. 4-9 presents the evolution of the metrics with the number of iterations for the Iterative Decentralized Algorithm. The result is clear: the gain of increasing the number of iterations is counterbalanced by the time it takes to perform the loops. This result justifies *a posteriori* to perform only 1 iteration for this algorithm.

4.6.4 Algorithms Comparison for Fixed Time Step

As was said, in the simulations presented in this section, the time step was set to 1 second for all algorithms. The rationale for this decision is that, in real application, it is likely that the navigation algorithm is not the only application requiring computations. Thus, if for instance a control algorithm is to be used, the computing time has to be shared, and only a percentage of the time is dedicated to the navigation algorithm.

In this scenario, the "pure" performances of the algorithms are compared. This means that, contrary to the varying time step scenario, the time needed to perform the computations is not taken into account in comparing the accuracies, here. The algorithms take the time needed to complete the computations, and the final estimates are compared. The results are presented on Fig. 4-10.

The Centralized Algorithm performs very well, as expected. Indeed, all information is provided to the Master, which can use it to estimate in the best possible way the state of the fleet. However, unexpectedly, the Hierarchic Architecture has a very good accuracy. The Decentralized Algorithm is sub-optimal (around 30% less accurate than the Centralized Algorithm), as expected. The Iterative Decentralized Algorithm performs almost as well as the Decentralized Algorithm, which is expected when the time to compute is not taken into account.

4.7 Conclusion

This study confirmed most of the expected results.

The Centralized Algorithm is optimal, but needs to perform heavy computations, especially for large fleets. As a result, the real-time accuracy is rather poor when the time-step is not fixed. Its synchronization and communication metrics are relatively low.

The Decentralized Algorithm is suboptimal, but rather close to the Centralized

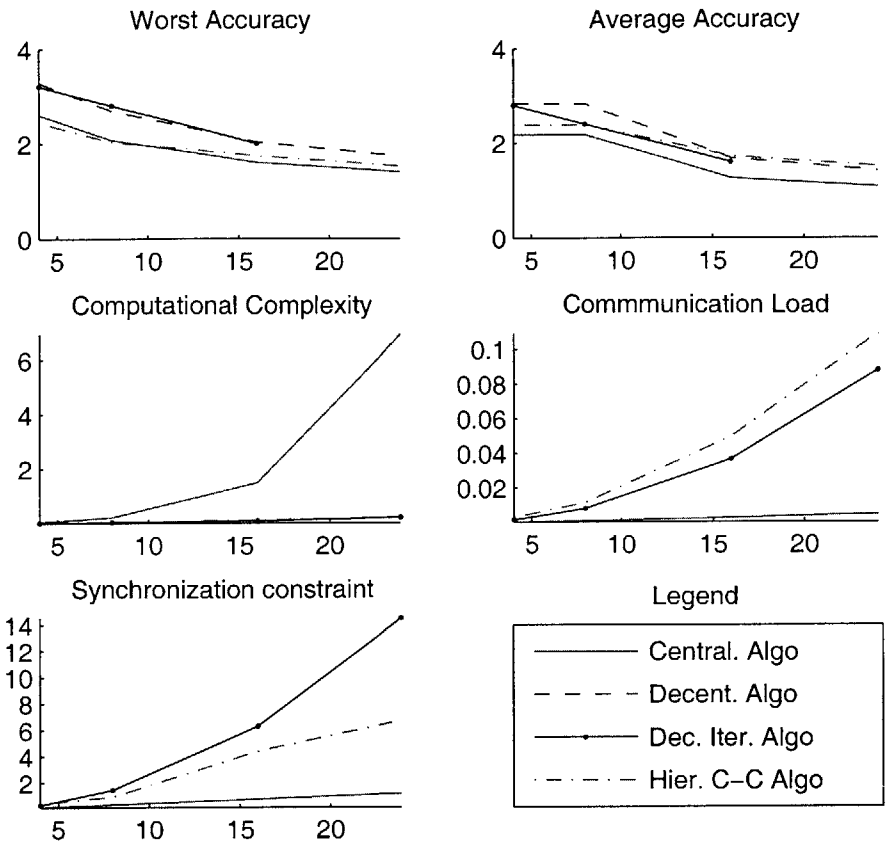

Figure 4-10: Comparison between the algorithms for fixed time step scheme

Algorithm in terms of accuracy. The computational burden being much lower than for the latter, the real-time accuracy is fairly good for the Decentralized Algorithm. The synchronization and communication metrics are fairly high for this algorithm, which would be its main flaws.

The Hierarchic Algorithm performs well in terms of accuracy. The synchronization level is rather low. The communication load is high, and this is due to cross-clusters communications, which are an essential part of the algorithm.

Finally, the Iterative Decentralized Algorithm show poor performances because of the loop over the whole fleet it needs to perform. The communication and synchronization metrics are fairly high.

Further studies could analyze more precisely some refinements of the Hierarchic Scheme for particular scenarios of special interest. In particular, the number of Sub-Clusters could be chosen more carefully for each scenario of interest. Moreover, the notion of robustness, which is particularly delicate to handle, could be analyzed to get insight into the capability of every algorithms to resist all kinds of failures.

Chapter 5

Comparing Three Strategies to Process Delayed Measurements

5.1 Introduction

In the context of the Magnetospheric MultiScale (MMS) mission, four satellites flying in formation estimate their positions based on GPS and crosslink range measurements. Previous work [6, 9] analyzed different position estimation algorithms for this mission, and raised a particular issue: in the actual mission, some information needed to perform the estimation arrived with a large time delay. This section analyzes different approaches to process this delayed information.

The problem analyzed is the following: four satellites flying in formation estimate their position. Each satellite receives GPS updates on its position, and crosslink range measurements with respect to every other satellite. Besides, every satellite sends its own position estimate regularly to the others, so that they can use it together with the crosslink range measurements to improve their position estimate. However, this estimate arrives with a time delay, which is modeled as constant. On a fast time scale (every T_f seconds), every satellite receives GPS and crosslink range measurement updates. On a slow time scale (every $T_s > T_f$ seconds), every satellite receives

updates on what the estimated positions of the other satellites were T_s seconds ago. The problem studied in this chapter is to process this delayed information efficiently, optimizing three criteria: 1) the real-time accuracy of the estimation process; 2) the complexity of the computations to perform; and 3) the memory required to run the algorithm.

A straight-forward way to process delayed information that arrives at time t is to store all the measurements between the last arrival of this update -$(t - T_s)$- and now, and to process them all from that time to current time. Based on this idea, one approach would be to implement a simple algorithm on a fast time scale - strictly a prediction algorithm, that does not process measurements-, while storing all the GPS and crosslink range measurement. Then, upon arrival of the delayed update, the algorithm goes back to the last update and processes all the measurements up to current time. This is the algorithm called Approach 1 in this chapter. However, in this approach, the real-time accuracy is very poor, since the measurement are not processed in real-time. A second approach (Approach 2) processes the GPS and crosslink range measurements on the fast time scale and, on a slow time scale, upon arrival of the delayed update, re-processes them. This approach provides the best accuracy of the three approaches, but requires heavy computations and data storage. Finally, a third approach is proposed, aimed at reducing the complexity of Approach 2 while remaining close to its accuracy. In Approach 3, the GPS and crosslink range measurements are processed in real-time, and they are not stored. Upon arrival of the update on the other satellites positions, this information is propagated to current time. Then, it is blended with the estimate from the fast time scale algorithm described in Approach 2.

The structure of this chapter is as follows. The first part of the chapter presents the problem and the three approaches proposed. The second part consists of an analytical study of the three approaches in terms of accuracy, computational complexity and memory requirements. The third part presents the results of a simulation which supports the analytical study. A summary table and plot concludes the study.

5.2 Problem Statement

In this section, the problem is presented in mathematical terms. The dynamic model for the satellites is introduced, as well as the measurements available to every satellite. The core algorithm used to process these measurements is presented. Then, the three approaches proposed to process the delayed measurements are described.

5.2.1 Problem Description

$N = 4$ satellites are estimating their positions and velocities. The same absolute frame is used for all satellites (Frame A described in Chapter 2). GPS measurements provide each satellite with an estimate of its position in this frame. For a precise description of the spacecraft dynamics, a nonlinear model should have been used (see Section 2.6 for instance). However, for this analysis, a more simplified model, relying on the Linear Time Invariant assumption (see Section 2.4) was chosen. The satellite dynamics in continuous time are:

$$\frac{dx}{dt}(t) = Fx(t) + w(t) \tag{5.1}$$

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \dot{x_1}(t) \\ \dot{x_2}(t) \\ \dot{x_3}(t) \end{bmatrix}, \qquad F = \begin{bmatrix} 0_3 & I_3 \\ 0_3 & 0_3 \end{bmatrix}, \tag{5.2}$$

where $x(t)$ is the state of the considered satellite (position and velocity), $w(t)$ is the process noise, of spectral density:

$$Q(t) = \begin{bmatrix} 0 & 0 \\ 0 & \bar{Q}(t) \end{bmatrix}$$

The structure of $Q(t)$ means that the noise is an acceleration noise (only the velocity of the satellites is directly affected by this noise: the first three components of w are always zero). When written in discrete time, the equation becomes

$$x(t+T_f) = \Phi(t, t+T_f)x(t) + \tilde{w}(t),$$

where $\tilde{w}(t)$ has discrete time covariance Q_t

$$Q_t = e^{FT_f} \int_0^{T_f} e^{-F\tau} Q(\tau) e^{-F^T\tau}\, d\tau e^{F^T T_f} = \begin{bmatrix} 0 & 0 \\ 0 & \int_0^{T_f} \bar{Q}(\tau)\, d\tau \end{bmatrix}.$$

Φ is given by

$$\Phi(t, t+T_f) = e^{FT_f} = \begin{bmatrix} I_3 & T_f I_3 \\ 0_3 & I_3 \end{bmatrix}.$$

Note: later, the transition matrix $\Phi(t, t+T_f)$ will be written Φ for simplicity.

The notation x represents the local satellite state (position and velocity), while the notation y represents the remote satellites states. Thus, the size of x is 6, and the size of y is $6(N-1)$. Every satellite receives (see Fig. 5-1)

- Every T_f seconds (fast period), its range with respect to every other satellite, and a coarse GPS measurement on its 3-D position. In the simulations performed, $T_f = 1sec$, the covariance of the GPS measurements was $100m^2$ (precision of $10m$), and the covariance of the crosslink range measurements was $0.01m^2$ (precision of $10cm$). The crosslink range measurements available at satellite i of state

$$x_i(t) = \begin{bmatrix} x_i^1(t) \\ x_i^2(t) \\ x_i^3(t) \\ \dot{x}_i^1(t) \\ \dot{x}_i^2(t) \\ \dot{x}_i^3(t) \end{bmatrix}$$

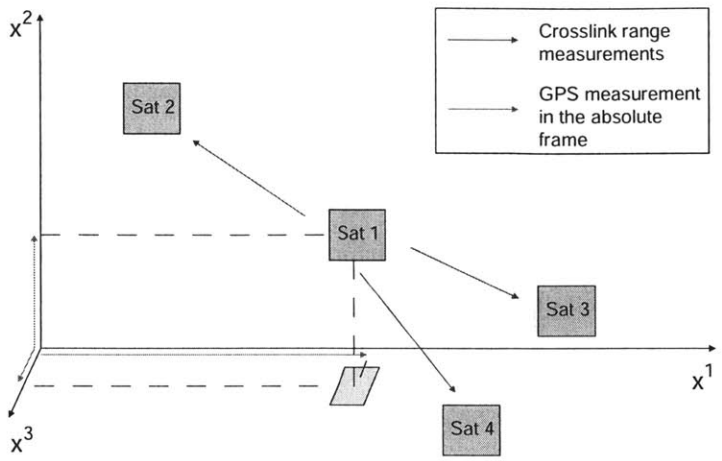

Figure 5-1: Sketch of the measurements available at a given satellite

are

$$r_{ij}(t) = \sqrt{\sum_{k=1}^{3} \left(x_i^k(t) - x_j^k(t)\right)^2} + v_{ij}(t),$$

where $v_{ij}(t)$ is the range measurement noise. The GPS measurements available at satellite i are

$$z_i(t) = \begin{bmatrix} x_i^1(t) \\ x_i^2(t) \\ x_i^3(t) \end{bmatrix} + \begin{bmatrix} v_i^1(t) \\ v_i^2(t) \\ v_i^3(t) \end{bmatrix},$$

where $v_i^k(t)$ are the GPS measurement noises, assumed to be independent and white [1].

- Every T_s seconds (slow period), an update on where the other satellites are *estimated* to be T_s seconds ago, together with the related covariance attributed to their estimate. That is, at time t, every satellite i receives an update on the states of the other satellites at time $(t - T_s)$: $\hat{y}_j(t - T_s), \forall j \neq i$.

The problem analyzed here is: how to process these delayed updates on other satellites states estimates ?

[1] GPS is considered here as a "black box" providing such a measurement.

5.2.2 Reduced-Order Filter

This part presents the core algorithm used to process the measurements. The goal of the algorithms is to compute the position and velocity of the local satellite only: each satellite is not directly interested in computing the position of the other satellites (for complexity reduction purpose). The algorithms are based on the Schmidt-Kalman Filter (SKF). Again, in the following equations, $\hat{x}$ represents the local satellite state estimate -position and velocity. P_{xx} is the related covariance of the local satellite's state estimate. For every remote satellite i, $\hat{y}_i$ represents its state estimate, $P_{yy,i}$ the related covariance. $\hat{y}$ represents the concatenation of all the remote satellites states estimates $\hat{y}_i$ together, and P_{yy} of the covariances. Every satellite maintains an estimate on its state, $\hat{x}$ and on the others' states, $\hat{y}_i$, together with a global covariance matrix

$$P = \begin{bmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{bmatrix}$$

The baseline for the three approaches is the Schmidt-Kalman Filter, which takes place in two steps, repeated over time

- Time Update. From one short time-step to the other, every satellite propagates its local state (position and velocity), and the state of the remote satellites, through $x(t+1) = \Phi x(t)$ and $P_{xx}^{+} = \Phi P_{xx}^{-} \Phi^T + Q_t$ for the local satellite, and $y_i(t+1) = \Phi y_i(t)$ and $P_{yy,i}^{+} = \Phi P_{yy,i}^{-} \Phi^T + Q_t$ for every remote satellite.

- Measurement Update. To process the range and GPS measurements, the local satellite's state and the covariances are modified by

$$\begin{aligned} \alpha &= H P_{xx}^{-} H^T + H P_{xy}^{-} J^T + J P_{yx}^{-} H^T + J P_{yy}^{-} J^T + R \\ &= \begin{bmatrix} H & J \end{bmatrix} P \begin{bmatrix} H^T \\ J^T \end{bmatrix} + R \qquad (5.3) \\ K &= \left(P_{xx}^{-} H^T + P_{xy}^{-} J^T \right) \alpha^{-1} \qquad (5.4) \\ x^{+} &= x^{-} + K(z - \hat{z}) \qquad (5.5) \end{aligned}$$

$$P_{xx}^{+} = (I - KH)\,P_{xx}^{-} - KJP_{yx}^{-} \tag{5.6}$$

$$P_{xy}^{+} = (I - KH)\,P_{xy}^{-} - KJP_{yy}^{-} \tag{5.7}$$

$$P_{yx} = P_{xy}^{T} \tag{5.8}$$

$$P_{yy}^{+} = P_{yy}^{-} \tag{5.9}$$

In these equations, z is the measurement, which can be written as a function of the states $z = f(x, y)$. $\hat{z}$ is the predicted measurement $\hat{z} = f(\hat{x}, \hat{y})$. $H = \left(\frac{\partial f}{\partial x}\right)_{\hat{x},\hat{y}}$ and $J = \left(\frac{\partial f}{\partial y}\right)_{\hat{x},\hat{y}}$.

The states of the remote satellites are not modified; the measurements are used to update the local state only. Note that the knowledge of the other satellites positions is included into $\hat{z}$ and the knowledge of their covariance and of the cross-correlations between local and remote satellites is included into K. During the covariance update, only the local covariance, and the cross-correlations between local and remote satellites are updated. The covariance of the other satellites is not modified during the measurement update step.

Again, the measurements available at a satellite of state estimate $\hat{x}$ are used to update this satellite's state estimate $\hat{x}$ only; they are not used to update the state of the remote satellites, $\hat{y}$.

The effect of the reduced-order assumption defined in the previous paragraph on the accuracy/complexity will be studied later in this chapter.

5.2.3 Description of the Three Approaches

In this section, some general equations used in the approaches, and then the three approaches are presented. Some equations used in the different approaches are as follows. The prediction equations for local satellite between time t and time $t + 1$

$$\begin{aligned} \hat{x}(t+1) &= \Phi\hat{x}(t) + a(t) \\ P_{xx}(t+1) &= \Phi P_{xx}(t)\Phi^{T} + Q \end{aligned} \tag{5.10}$$

In these equations, $a(t)$ is the acceleration undergone by the considered satellite at time t. For remote satellites

$$\begin{aligned} \hat{y}_i(t+1) &= \Phi\hat{y}_i(t) + a_i(t), \forall i \\ P_{yy,i}(t+1) &= \Phi P_{yy,i}(t)\Phi^T + Q, \forall i \end{aligned} \tag{5.11}$$

Measurement Update Equations at time t (for GPS and crosslink range measurements)

$$\begin{aligned} \alpha &= HP_{xx}^- H^T + HP_{xy}^- J^T + JP_{yx}^- H^T + JP_{yy}^- J^T + R \\ &= \begin{bmatrix} H & J \end{bmatrix} P \begin{bmatrix} H^T \\ J^T \end{bmatrix} + R \\ K &= \left(P_{xx}^- H^T + P_{xy}^- J^T\right)\alpha^{-1} \\ x^+ &= x^- + K(z - \hat{z}) \\ P_{xx}^+ &= (I - KH)\,P_{xx}^- - KJP_{yx}^- \\ P_{xy}^+ &= (I - KH)\,P_{xy}^- - KJP_{yy}^- \\ P_{yx} &= P_{xy}^T \\ P_{yy}^+ &= P_{yy}^- \end{aligned} \tag{5.12}$$

Update equations for remote satellites (slow measurement)

$$\begin{aligned} y_i &= y_i^{updated}, \forall i \\ P_{yy,i} &= P_{yy,i}^{updated}, \forall i \end{aligned} \tag{5.13}$$

The three approaches are as follows

1. Approach 1

   - Fast Time-Scale: a simple prediction is implemented. Every satellite propagates its local state and covariance according to Equations 5.10, and the other satellites states and covariances according to Equations 5.11.

- Slow Time-Scale: a batch processing is implemented. The algorithm comes back to T_s seconds before current time, uses the update for the other satellites states and covariances (Equation 5.13), and processes all the measurements from that time to current time (Equations 5.10, 5.11, 5.12). Note that all the measurements (range and GPS) on the period $[(n-1)T_s; nT_s]$ had to be stored. At time $(n-1)T_s$, the update for other satellite is performed (slow measurement) according to Equations 5.13. Then, for every time between $(n-1)T_s$ and nT_s, the prediction equations are used to propagate the local satellite state and covariance (Equations 5.10) and the remote satellites states and covariances (Equations 5.11), and the measurement update equations (Equations 5.12) are used to process the measurements. Note that the Kalman gain is computed here at each time step according to Equation 5.4. All the measurements in $[(n-1)T_s; nT_s]$ had to be stored.

2. Approach 2:

   - Fast Time-Scale: a SKF is implemented in the way described in the previous section. The state of the remote satellite is not directly estimated, except through simple prediction. The error on the remote satellites states grows. Equations 5.10 and 5.11 are used to compute the prediction step, and Equations 5.12 are used for the measurement update step.
   - Slow Time-Scale: a batch processing is implemented, as for Approach 1. At time $(n-1)T_s$, the update for other satellite is performed (slow measurement) according to Equations 5.13. Then for every time between $(n-1)T_s$ and nT_s, the same two sets of operations are performed: 1) prediction step, by use of Equations 5.10 and 5.11, and 2) measurement update step, by use of Equations 5.12.

3. Approach 3:

   - Fast Time-Scale: a SKF is implemented as for Approach 2. The prediction step uses Equations 5.10 and 5.11, and the measurement update step uses Equations 5.12.

   - Slow Time-Scale: first, the algorithm comes back to the state at time $(n-1)T_s$. The updates on the other satellites states and covariances are included through Equations 5.13. Then, this estimate is propagated all the way from $(n-1)T_s$ to nT_s, *without including the range and GPS measurements* taken at the faster time scale (which were not stored), by use of Equations 5.10 and 5.11 for all time between $(n-1)T_s$ and nT_s. Contrary to the batch processing performed in Approach 1 and 2, this approach uses a mere prediction, without measurement update. There is no need to store the measurements between time $(n-1)T_s$ and nT_s. Then, this estimate from the long time-step filter $\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}$ and the estimate from the short time-step filter (STF), $\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$, are blended according to the following equations (see derivation of these equations in Appendix A.1):

$$\begin{bmatrix} \hat{x}^+ \\ \hat{y}^+ \end{bmatrix} = \begin{bmatrix} \hat{x} + P_{xy}\left(I - P_{yy}^{-1}\tilde{P}_{yy}\right)P_{yy}^{-1}(\tilde{y}-\hat{y}) \\ \tilde{y} \end{bmatrix}$$

$$P^+ = \begin{bmatrix} P_{xx} - P_{xy}\left(I - P_{yy}^{-1}\tilde{P}_{yy}\right)P_{yy}^{-1}P_{yx} & P_{xy}P_{yy}^{-1}\tilde{P}_{yy} \\ \tilde{P}_{yy}P_{yy}^{-1}P_{yx} & \tilde{P}_{yy} \end{bmatrix} \quad (5.14)$$

5.3 Analytical Study

In this section, the three approaches are analyzed. Each section presents the analysis for one approach: first, in terms of accuracy, analyzing how it compares with other approaches. Second, in terms of complexity. Third, in terms of memory requirements.

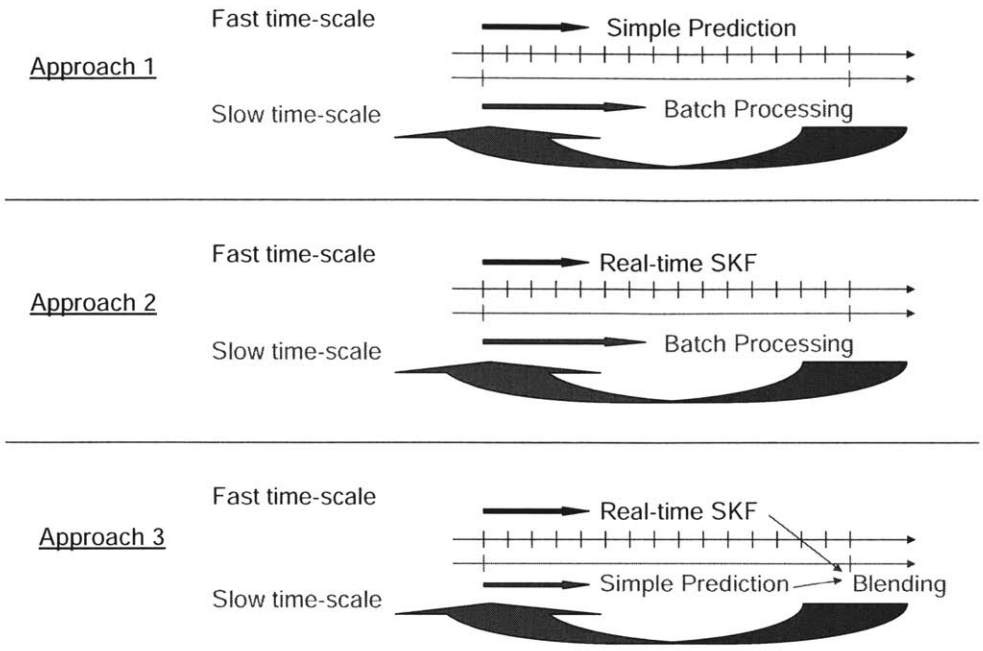

Figure 5-2: Sketch for the three approaches

For each approach, a brief summary presents the main strengths and weaknesses.

5.3.1 Approach 1: Prediction / Batch Processing

In Approach 1, a simple prediction is performed on the fast time-scale, and a batch processing on the slow time-scale.

Accuracy

The error keeps growing since a mere propagation is performed. The error at the end of a slow period is given by the integration of the Riccati Equation over this period of time:

$$P^+ = \Phi P \Phi^T + Q,$$

where Φ is the transition matrix and Q is the process noise covariance. The filter is optimal, since it processes all the measurements between two slow measurements.

To get insight into the accuracy evolution over the fast time-scale in this approach, one would like to solve the corresponding Riccati equations. However, this equation corresponds to the blending of T_s Riccati equations for a nonlinear model. It is thus a very hard equation. Thus, a simplified model is used. This simplified model is a linear model -to be able to solve it- which is close to the problem discussed in this chapter. This simplified model consists of two one-dimensional vehicles estimating their state (1-D position and velocity). On the fast time-scale (1Hz), they get a measurement of their position x_i (GPS-like) and the difference between their positions $r = x_2 - x_1$ (a linear measurement similar to the crosslink range measurement). On the slow time-scale (every T_s seconds), they get an estimate of what was the other's position T_s seconds ago. The propagation equation is:

$$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}_t + \begin{bmatrix} 0 \\ w \end{bmatrix},$$

where w is of covariance q. The covariance growth is given by

$$P^+ = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} P \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}.$$

Solving this equation over the slow period leads to

$$P = \begin{bmatrix} \tilde{a} & \tilde{b} \\ \tilde{b} & \tilde{c} \end{bmatrix},$$

with:

$$\begin{aligned} \tilde{a} &= \left((2b_m + q/6)T_s^3 + (c_m - q/2)T_s^2 + (q/3)T_s + a_m\right), \\ \tilde{b} &= \left((q/2)T_s^2 + (c_m - q/2)T_s + b_m\right), \\ \tilde{c} &= \left(qT_s + c_m\right), \end{aligned}$$

where

$$P_m = \begin{bmatrix} a_m & b_m \\ b_m & c_m \end{bmatrix}$$

is the covariance matrix at the beginning of the slow period. The error on the position grows linearly with the process noise covariance q, and much faster with the slow period ($P_{11} = \alpha T_s^3$). This gives an idea of the drift of the error over the slow period for this approach.

Complexity

Fast time-scale: the complexity of a prediction step for each satellite is derived in Appendix A.2. This complexity is 510 operation counts per satellite. For 4 satellites, the complexity of a prediction step is: 2040.

Slow time-scale: on the slow time-scale, the batch processing implies performing prediction and update at each step back to the last slow measurement. The complexity of an update step is derived in Appendix A.2: it is 9414 operation counts. Over a slow period, the total complexity is then: $(2040 + 9414)T_s = 11454T_s$.

Memory requirement

The batch processing needs to save all the fast measurements between two slow updates. This consists of $(N-1)$ range data, plus 3 GPS data, T_s times between two slow measurements. This is $(N+2)T_s$ data to save between two slow measurements.

Summary

This first approach is optimal but intensive on the slow time-scale. It is very coarse, though very cheap in terms of computations.

Pros:

- Cheap on the fast time-scale;
- Optimal on the slow time-scale.

Cons:

- Poor accuracy on the fast time-scale;
- Computationally hard on the slow time-scale;
- Huge memory requirement to complete the batch processing.

5.3.2 Approach 2: Filter / Batch Processing

In Approach 2, an Extended Kalman Filter is performed on the fast time-scale, and a batch processing on the slow time-scale.

Accuracy

The accuracy is determined by the SKF. More precisely, one can compute roughly the accuracy that is reached. Let us consider a symmetrical case, where all satellites have the same uncertainty, say P. Now, since the GPS is coarse, when P is low, it is determined mainly by the range measurements. With a fleet of 4 satellites, the position of the local one can be determined by the three crosslink ranges. The resulting covariance will be roughly the sum of the range covariance and of the other satellites covariance. thus, the error on the local satellite's position, p, will be roughly: $p^+ = \sqrt{r^2 + (p^-)^2}$. Thus, each fast time step, this error will grow by $\sqrt{r^2 + (p^-)^2} - p^-$. (see the simulation result section for a confirmation of this approximate law). This growth is to be compared with the growth in the prediction-only scheme used by Approach 1. Again, optimality is reached because of the batch processing, as for Approach 1.

Complexity

Fast time-scale: the complexity on the fast time-scale is the complexity of a prediction + update step, which is for $N = 4$, based on the study in the previous section, 11454.

Slow time-scale: on the slow time-scale, a batch processing is performed. The analysis is the same as for Approach 1: the complexity is $11454T_s$.

Memory requirement

The memory requirement is the same as for Approach 1.

Summary

This approach is optimal from every viewpoint (except that it uses the Schmidt-Kalman Filter, which is an approximate algorithm). It is also the most intensive in terms of computation and memory requirement.

Pros:

- Optimal on both slow and fast time-scale.

Cons:

- Computationally very hard, both on the slow and the fast time-scale;
- Huge memory requirement to complete the batch processing.

5.3.3 Approach 3: Filter / Filter+Blending

In Approach 3, an Extended Kalman Filter is performed on the fast time-scale. On the slow time-scale, a propagation is performed from the last slow measurement time, and these two estimations are blended.

Accuracy: Comparing Approach 2 and 3

In this section, Approach 2 and 3 are compared in terms of their accuracies. The goal is to build a simplified model that accounts for their differences, while being solvable exactly. This simplified model will consist of two algorithms estimating the state of a one-dimensional system. Algorithm A is a simplified version of Algorithm 2, and Algorithm B is a simplified version of Algorithm 3. Let us first note the difference between Approach 2 and 3: the only difference between them is that Approach 2 uses the update on the remote satellites at the time it was sent (through the batch

processing, which goes back to the last slow update to process the update at the time it was sent), whereas Approach 3 uses this update at the time it is received. The simplified model is now built by comparing Approach 2 and 3 in two steps

1. In the first step, Approach 3 is compared with the batch processing performed in Approach 2, *as if it was performed in real-time*. The first part of the simplified model is built out of this comparison.
2. In the second step, the short time-step filter is introduced for Approach 2, and the difference in accuracy between both approaches is studied. The second part of the model is made out of this second step.

First Step. Note that Approach 3 and the batch processing of Approach 2 use the same measurements, the same information, apart from the estimates for the remote satellites. Indeed, on the time-lag $[nT_s; (n+1)T_s]$, both algorithms update the estimates for the remote satellites from the slow measurements, and then use all the fast measurements until time $(n+1)T_s$ with the same SKF. Only the accuracy of the estimates of the remote satellites is different: Approach 3 uses outdated estimates, while the batch processing uses updated ones. The estimates on the remote satellites states are used only to process the range measurements. As a result, a different accuracy for the remote satellites estimates has the same effect as a difference in the measurement accuracy. Thus, the simplified model is built by considering Approach 3 and the batch processing as being the same algorithm applied to measurements of different accuracies. More precisely, Approach 3 uses outdated estimates for the remote satellites. Then, the covariances of these estimates is increased by the process noise covariance. Since the covariance measurement is roughly $R + P_{remote}$ (R is the nominal covariance of the measurement, and P_{remote} is the covariance of the considered remote satellite state estimate), Approach 2 and 3 use measurements whose covariances differ by:

$$(R + P_{remote,outdated}) - (R + P_{remote,updated}) = \Delta P = Q$$

Indeed, since Approach 3 uses outdated estimates for the remote satellites, their covariances have to be increased by the process noise covariance. Thus, it is as if the process noise covariance was added to the measurement covariance. In our simplified model, the batch processing would be using measurements of covariance R while Approach 3 would be using measurements of covariance $R+Q$. At steady-state, the covariance P is given by the equality between the covariance before and after the Prediction and Measurement Update steps

$$P_A = \frac{(P_A+Q)R}{P_A+Q+R} \Rightarrow P_A = \frac{Q}{2}\left(-1+\sqrt{1+\frac{4R}{Q}}\right),$$

for Algorithm A, and

$$P_B = \frac{(P_B+Q)(R+Q)}{P_B+2Q+R} \Rightarrow P_B = \frac{Q}{2}\left(-1+\sqrt{5+\frac{4R}{Q}}\right),$$

for Algorithm B. Then

$$\Delta P = P_B - P_A = \frac{Q}{2}\left(\sqrt{5+\frac{4R}{Q}} - \sqrt{1+\frac{4R}{Q}}\right).$$

Second Step. Now, the difference between Approach 2 and 3 in real-time is not ΔP since the batch processing is not performed in real-time, it is only performed every T_s seconds. In real-time, Approach 2 performs the short time-step filter. Note that from nT_s+1 (right after the batch processing) and $(n+1)T_s-1$ (right before the next batch processing), Approach 2 and 3 perform exactly the same operations -the SKF- with the same measurements, and the same estimation for the remote satellites (they have been updated for both approaches). To analyze what happens in the time-lag $[nT+1;(n+1)T-1]$, another time step is added to our simplified model, when steady-state is reached. In this additional time-step, Algorithm A -which represents Approach 2- uses measurements of same accuracy as Algorithm B -representing Approach 3-: the covariance of these measurements is $R+Q$. This additional step leads to the covariances

$$P_A^+ = \frac{(P_A+Q)(R+Q)}{P_A+2Q+R},$$

$$P_B^+ = \frac{(P_B+Q)(R+Q)}{P_B+2Q+R} = P_B.$$

The new difference in the covariances corresponds to the difference of accuracies between Approach 2 and 3 at time $nT_s - 1$ (right before the batch processing). It is

$$\begin{aligned}\Delta P^+ = P_B^+ - P_A^+ &= \frac{(P_B+Q)(R+Q)}{P_B+2Q+R} - \frac{(P_A+Q)(R+Q)}{P_A+2Q+R} \\ &= \frac{(R+Q)^2\Delta P}{(P_A+2Q+R)(P_B+2Q+R)} < \Delta P.\end{aligned}$$

Now

- If $Q \ll R$, then $\Delta P \approx 0$, and thus $\Delta P^+ \approx 0$.
- If $Q \gg R$, then $\Delta P \approx 0.6Q$, and $\Delta P^+ \approx \frac{\Delta P}{4} \approx \frac{Q}{7}$.

This analysis shows that, when the process noise covariance is low as compared with the measurement covariance, Approach 2 and 3 lead to similar results. When the process noise covariance is higher, their difference is higher too. This result has a straight-forward interpretation: the main difference between Approach 2 and 3 is that Approach 3 uses the same information (the estimate on the remote satellites states), but later (the update on the remote satellites states is used at the next slow period for Approach 3, whereas Approach 2 comes back to the last slow period to use the update). The value of such a delayed information depends on the process noise covariance: when this noise is high, the covariance of the remote satellites estimates grows very quickly, making it less useful. On the contrary, when the process noise is low (compared to the measurement covariance), the update on the remote satellites can be used later (at the next slow period) without much loss. As a result, when the process noise covariance is low, using the update on the remote satellites does not change much, and Approach 2 and 3 are equivalent. When the process noise covariance is high, the difference is much more visible. With the process noise covariance used in the simulations, both approaches showed very similar accuracies, confirming this analysis.

Complexity

Fast time-scale: the complexity on the fast time-scale is the same as for Approach 2: 11454.

Slow time-scale: on the slow time-scale, the prediction over the slow period is of complexity $2040T_s$. The complexity of the blending phase, derived in Appendix A.2, is 6666. Thus, the total complexity on the slow time-scale is thus: $6666 + 2040T_s$. Note that the complexity of the blending phase does not depend on the slow period, contrary to the complexity of the slow period in Approach 1 and 2. When $T > 8$, the complexity of Approach 1 or 2 on the slow period is four times more than the complexity of Approach 3.

Memory requirement

No measurement has to be saved for this approach. Each measurement -the fast measurements and the slow ones- are used when they arrive, so that they do not need to be stored.

Summary

This approach is close to be optimal. Moreover, it is much less intensive (in terms of computation and memory) than Approach 2.

Pros:

- Close to optimality,
- Computationally less intensive than Approach 2,
- Memory requirement very low (no batch processing).

Cons:

- Sub-optimal estimate on the slow time-scale.

5.3.4 Analytical Comparison Summary

In this section, the analytical study performed in the previous sections is summarized. The effects of the main parameters on the conclusions are emphasized.

On the fast time scale, Approach 1 predicts the state of the satellites, while Approach 2 and 3 use the GPS and range measurements to improve this estimate. The analytical study, based on simplified model, showed that the covariance in Approach 1 grows linearly with the noise covariance, and as the power 3 of the time elapsed. The covariance in Approach 2 and 3 grows by the range measurement covariance at each time step. For a large slow period (large time delay for the update of remote satellites states), the prediction scheme of Approach 1 is drifting rapidly (as the power 3 of this period) from the truth, whereas the filter of Approaches 2 and 3 drifts much slower (linearly with the period). For a large process noise (as compared with the measurement covariance), this drift is even faster.

On the slow time scale, Approach 1 and 2 use the batch processing of all data from the last update for remote satellites to current time, including all the measurements on this time window. Approach 3 predicts the state of the remote satellites over this time window, and merge this estimate with the estimate from the short time step filter. The analytical study of the simplified model showed that if the noise is low, or the slow period is short, then the difference between the two approaches is negligible. When the process noise becomes more important, the difference becomes important too.

5.3.5 Effect of Reduced-Order Assumption on Complexity

In this section, the effect on the complexity of implementing reduced-order algorithms is analyzed. The full-order Extended Kalman Filter would use the following equations

- Prediction Equations (same as the reduced-order algorithm). For local satellite

between time t and time $t+1$

$$\begin{aligned} \hat{x}(t+1) &= \Phi\hat{x}(t) \\ P_{xx}(t+1) &= \Phi P_{xx}(t)\Phi^T + Q \end{aligned} \tag{5.15}$$

For remote satellites

$$\begin{aligned} \hat{y}_i(t+1) &= \Phi\hat{y}_i(t), \forall i \\ P_{yy,i}(t+1) &= \Phi P_{yy,i}(t)\Phi^T + Q, \forall i \end{aligned} \tag{5.16}$$

- Measurement Update Equations at time t

$$\begin{aligned} K &= P^- H^T \left(HP^- H^T + R\right)^{-1} \\ \begin{bmatrix} x \\ y \end{bmatrix}^+ &= \begin{bmatrix} x \\ y \end{bmatrix}^- + K(z - \hat{z}) \\ P^+ &= (I - KH)\, P^- \end{aligned}$$

- Update Equations for remote satellites (same as the reduced-order algorithm)

$$\begin{aligned} y_i &= y_i^{updated}, \forall i \\ P_{yy,i} &= P_{yy,i}^{updated}, \forall i \end{aligned} \tag{5.17}$$

The difference in complexity would then be

- For Algorithm 1. There is no difference in complexity on the fast time-scale (prediction only - the same equations are used for reduced-order and full-order algorithms). On the slow time-scale, the complexity of the full-order algorithm is
  - Kalman Gain computation: $(6N)^2(N+2)$ for PH^T, $(6N)(N+2)^2$ to multiply it by H, $(N+2)^2$ to add R, $(N+2)^3$ to inverse, and finally

$(6N)(N+2)^2$ to multiply PH^T by $\left(HPH^T + R\right)^{-1}$.

- State Update: $(N+2)$ for $z - \hat{z}$, $(6N)(N+2)$ to multiply by K, $(6N)$ to add x^-.
- Covariance Update: $(6N)^2(N+2)$ for KH, $(6N)$ to subtract to I, and finally $(6N)^3$ to multiply by P^-.
- The total complexity for the update step is then: $(6N)^3+2(6N)^2(N+2)+2(6N)(N+2)^2+(N+2)^3+(N+2)^2+(6N)(N+2)+(N+2)+2(6N) = 22914$.

The complexity of the prediction and update is thus $22914+2040 = 24954$, more than twice the complexity for the reduced-order algorithm. As a result, the reduced-order filter may be the only possible solution when on-board computers are limited in the amount of computation they can perform.

5.4 Simulation Results

In this section, the results of the simulations performed are presented. They include the effect of the update from the remote satellites on the accuracy, the effect of the reduced-order assumption on the accuracy, and the comparison between the approaches itself.

5.4.1 Effect of the Update

In Appendix A.4, some figures are added to emphasize the gain of adding regularly an update on remote satellites states. Fig. A-2 shows the covariance and errors obtained with the update from other satellites (slow measurement). Fig. A-3 shows the same result when no update come from other satellites, while Fig. A-4 is the difference between these two graphs. The gain in doing this update increases in time, and stabilizes around 1m. Appendix A.3 explains why the effect of these updates is not as large as one might expect.

5.4.2 Effect of Reduced-Order Assumption on Accuracy

In this section, the comparison between the reduced-order and the full-order algorithms in terms of accuracy, based on simulation results is presented. Both sets of algorithms were implemented, with the same parameters, so as to compare the accuracy of the resulting estimations. Fig. 5-3 and 5-4 present the evolution of the covariance normalized by the covariance in the GPS only case (each satellite receives GPS updates, and does not receive crosslink range measurements). Fig. 5-3 represents the accuracy reached by the full-order algorithms, while Fig. 5-4 represents the accuracy reached by the reduced-order algorithms. The scales on the left and on the right side are the same. The scale for the first approach is $[0.5; 1.7]$, and for Approach 2 and 3 it is $[0.5; 1]$. As expected, using full-order algorithms improves the accuracy of the estimates (by $0.2m$ roughly for all algorithms). However, this gain in accuracy is to be balanced with the additional complexity of the full-order algorithms (see Section 5.4.2).

5.4.3 Comparison Between the Three Approaches

Fig. 5-5 represents the covariance evolution over time (normalized by the covariance in the GPS only scenario), and the complexity evolution over time for every approach. Fig. 5-6 presents these plots on the same figure, so as to make the comparison simpler.

The fast frequency is 1Hz, the slow frequency is 0.1Hz, and the total duration of the simulation is 100s. The conclusions of the analytical part are confirmed by the simulations:

- Approach 1. The error over a slow period is growing quickly (prediction only, going from 0.85 to 1.4, see Fig.5-6). At the end of the slow period the optimality is recovered through the batch processing (every $10s$ the curve corresponding to Approach 1 takes the same value as the curves corresponding to the other two approaches, around 0.85). The complexity is very low over the slow period ($3 \cdot 10^3$, see Fig. 5-5, but very high after each slow measurement ($12 \cdot 10^4$).

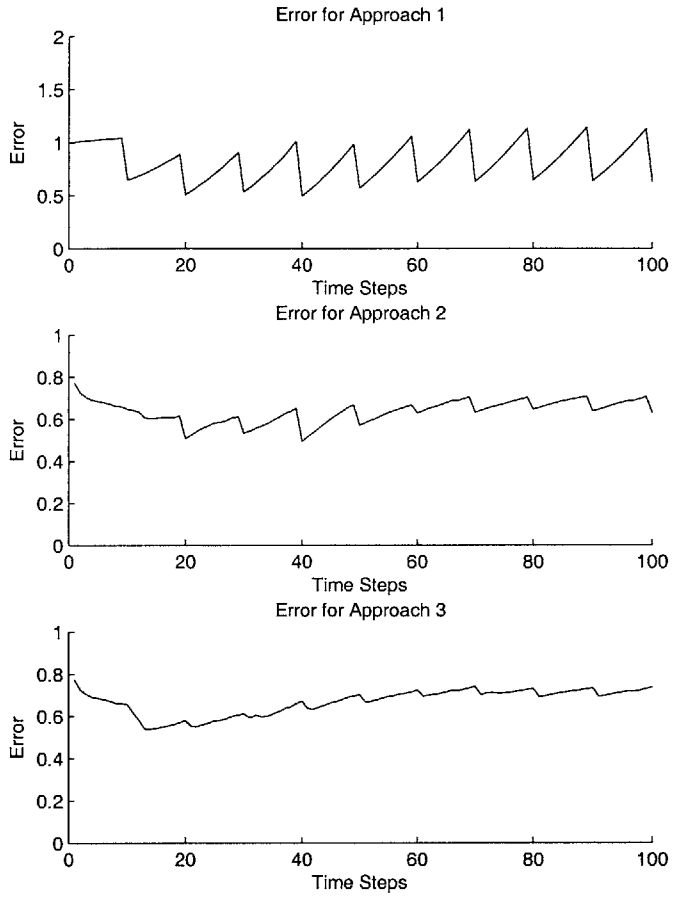

Figure 5-3: Covariance evolution for the three approaches with the full-order algorithms

- Approach 2. The error grows slowly over a slow period (between 0.85 and 0.95), since the algorithm processes the measurements in real-time (STF). However, it still grows since the estimation on the remote satellites is not updated in real-time. On the slow time-scale, because of the batch processing based on updated estimates for remote satellites, the estimation is also optimal (error around 0.85). The complexity of this approach is the highest on both fast (10^4) and slow ($12 \cdot 10^4$) time-scales.

- Approach 3. The error grows in the same fashion as Approach 2 on the fast

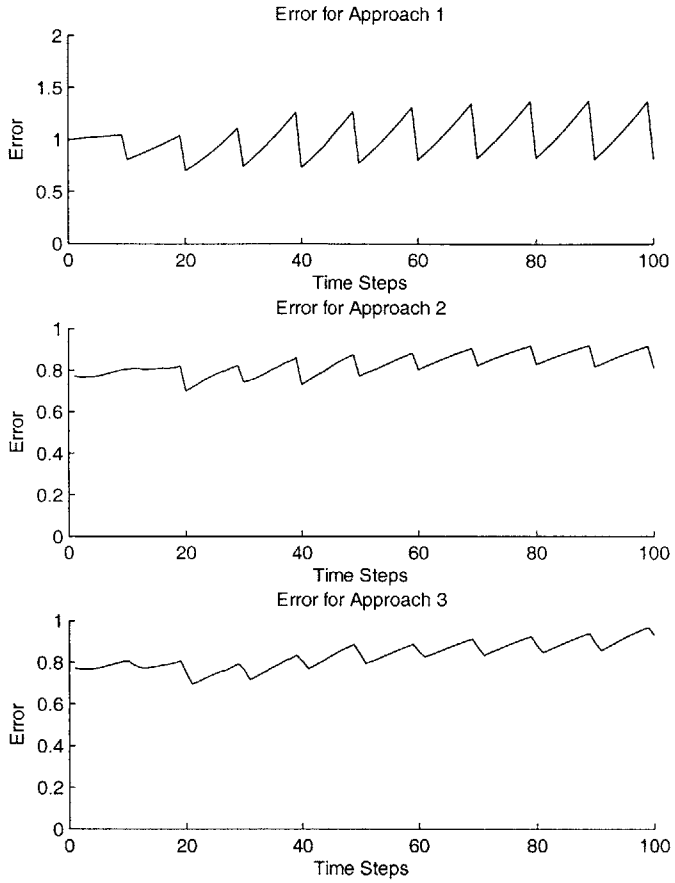

Figure 5-4: Covariance evolution for the three approaches with the reduced-order algorithms

time-scale (going from 0.85 to 0.95). On the slow time-scale, the error is closed to the error of Approach 2 or 3 (around 0.87, as compared with 0.85 for the other two approaches). The complexity is high on fast time-scale (10^4), and low on slow time-scale ($4 \cdot 10^4$), as compared with Approach 1 and 2.

5.5 Conclusion

As a result, it was shown that Approach 1 is very inaccurate on the fast time-scale, very intensive on the slow time-scale, and requires a significant amount of data to be stored. Approach 2 is optimal, but very computationally intensive, and requires

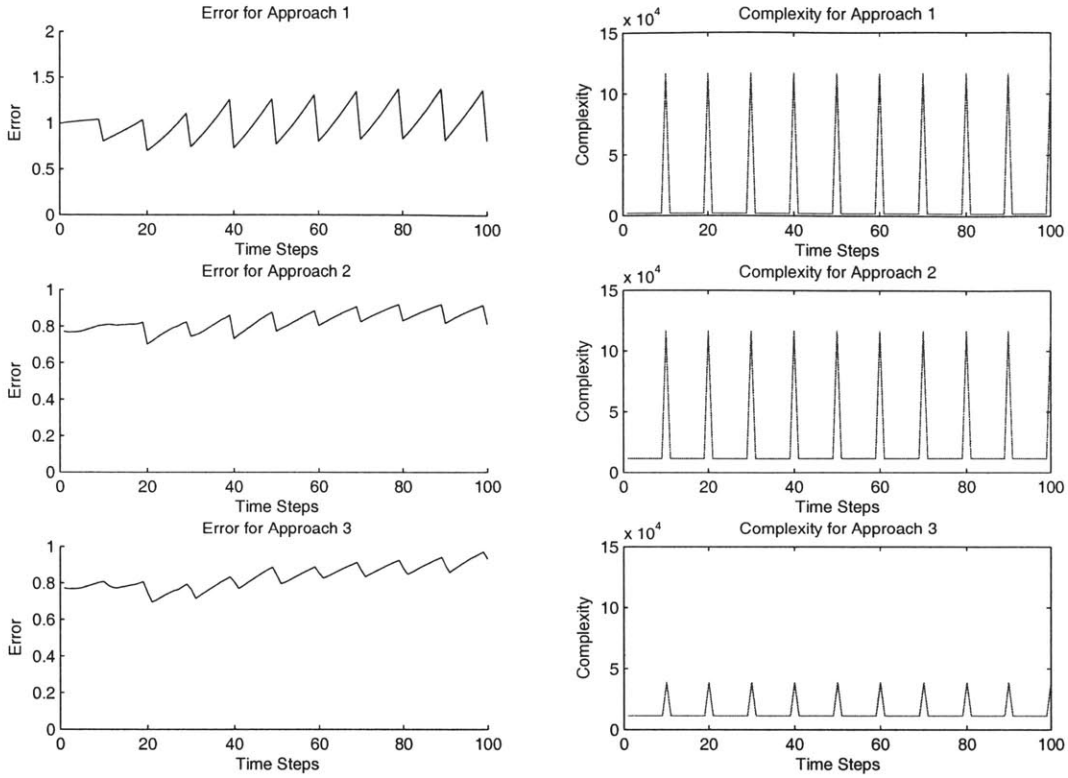

Figure 5-5: Comparison between the three approaches for the reduced-order algorithms (the covariances are normalized by the GPS-only estimation covariance)

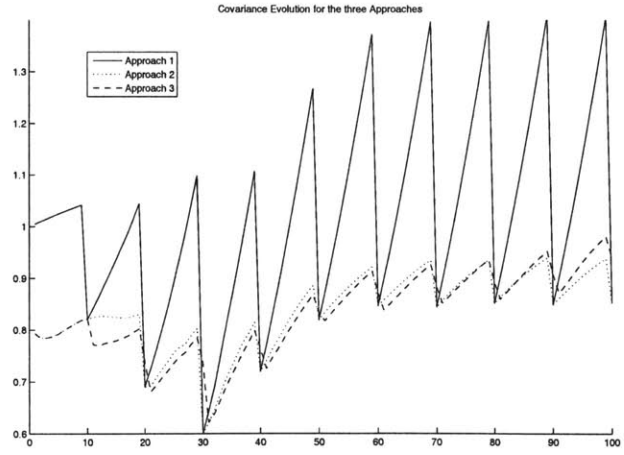

Figure 5-6: Comparison between the three approaches for the reduced-order algorithms (zoomed)

a huge amount of data to be stored. Finally, Approach 3 reaches almost the same accuracy as Approach 2, while being much less intensive in terms of computation ($4 \cdot 10^4$ operation count as compared with $12 \cdot 10^4$ for Approaches 1 and 2 on the slow time-scale). It does not need any data to be stored. These results are summarized in Table 5.1, by representing for each approach the strengths and weaknesses in terms of accuracy and complexity for fast and slow time-scales. Fig. 5-7, 5-8 and 5-9 represent summary complexity/accuracy plots.

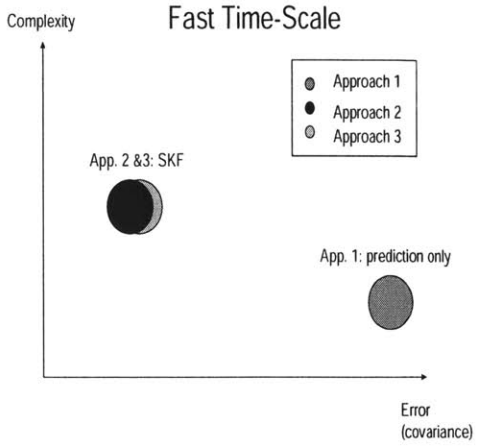

Figure 5-7: Complexity-accuracy graph for the three approaches on fast time-scale

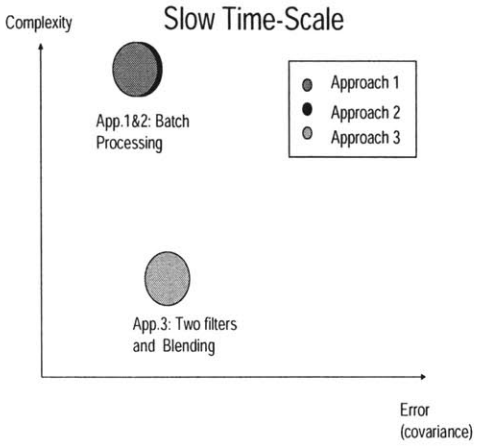

Figure 5-8: Complexity-accuracy graph for the three approaches on slow time-scale

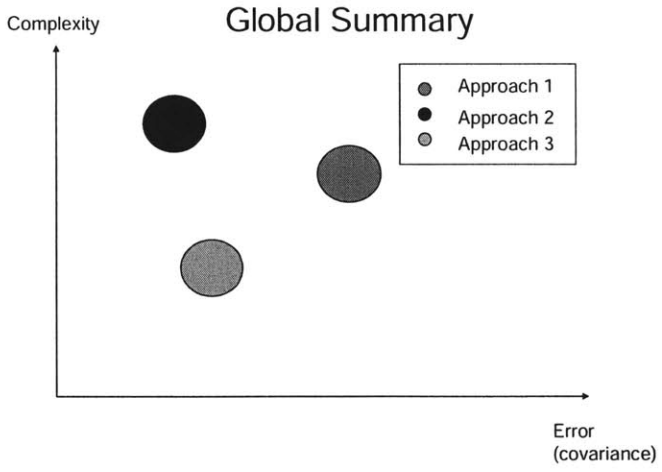

Figure 5-9: Summary complexity-accuracy graph for the three approaches

Table 5.1: Comparison Summary: strengths (+) and weaknesses (-) of each approach

	Fast		Slow	
App. Numb.	Accuracy	Compl. and Mem.	Accuracy	Compl. and Mem.
1	-	+	+	-
2	+	-	+	-
3	+	-	+	+

Chapter 6

Explanation of a Particular Instability of the EKF

6.1 Introduction

When the Extended Kalman Filter is used for a system involving sensors of very different accuracies, experience has shown that the estimation process can be unstable [37, 6]. In fact, when this accuracy ratio is very large, the estimate computed by the EKF almost never converges to the actual state of the system. This chapter analyzes this phenomenon and presents an approach to fix it.

6.2 The Reason for the Instability

6.2.1 Intuitive Explanation

Consider a system with two nonlinear sensors of very different accuracies. After the first measurement, the covariance matrix of the discrete filter is updated using [12]

$$P^+ = (I - KH)\,P^- = \left(I - P^- H^T \left(HP^- H^T + R\right)^{-1} H\right) P^-, \qquad (6.1)$$

where P^- denotes the covariance of the estimate before the measurement update, and P^+ the covariance after the update. During the first measurement update, the covariance matrix will get a very small eigenvalue in the direction associated with the more accurate measurement (because very accurate information is provided in this direction), but still have a large value in the direction of the coarse measurement (for a similar reason). Fig. 6-1 represents this first update step (large area): in one direction, the uncertainty ellipse is shrunk, corresponding to the direction of accurate measurement, and in the other direction, the uncertainty ellipse is wide, corresponding to the direction of coarse measurement. Now, during the second measurement update, the measurement matrix, which is the gradient of the measurement function, will have changed from the first step, because the state estimate has been updated during the first measurement update. More precisely, the two directions (accurate and coarse) associated with the second measurement update are not the same than the directions for the first update (due to a change in the estimate). The result is that during this second measurement update the uncertainty will shrink in a direction different from the accurate direction of the first measurement update. As a result, after this second measurement update, both directions are much reduced, and the uncertainty (covariance matrix) will be very small in every direction. The problem here is that this reduction of the covariance in every direction does not correspond to the true estimation process. It is due to errors in the linearization because the state estimate is not perfect. This error is the reason why the linearized measurement directions is moving. As a result, the filter acts as if the system was measured with an accurate sensor in two different directions, so that the covariance becomes very small in both directions. In reality, only one direction corresponds to accurate measurements, the other corresponding to coarse measurements. This intuitive explanation is represented on Fig. 6-1. The larger area represents the area of confidence after the first measurement update step, and the small area after the second measurement update step. On the left is represented the second update as it should be done, and on the right the second update as it actually occurs, with the "spillover" of the two directions.

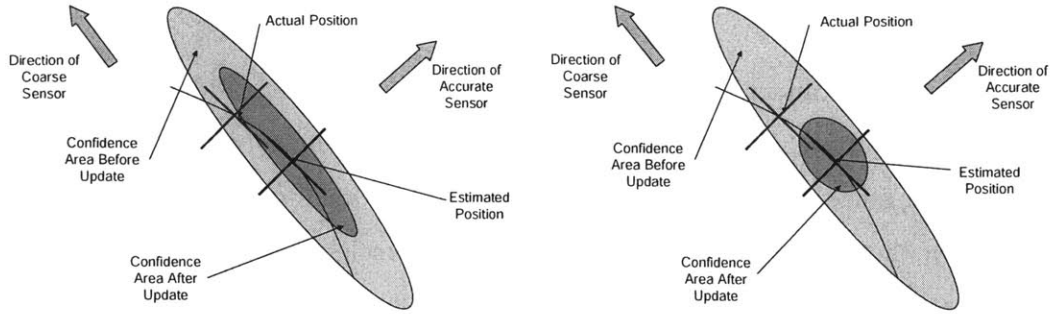

Figure 6-1: The Update Step as it should occur (left), and as it actually occurs (right). The line is the level line of the range measurement: on this line, the range is constant.

For this reason, the filter overestimates the confidence of the updated estimate in the direction of coarse measurement. Then, the subsequent measurements are considered relatively less accurate, and less taken into account. This prevents the state estimate from converging to the true state [12]. To summarize, the scenario by which the filter does not converge is the following:

1. During the first update step, the covariance matrix is shrunk in the direction of the accurate measurement. It is also reduced, but to a less extent, in the direction of the coarse measurement.
2. During the second step, the direction of the accurate measurement is modified because of the nonlinearity and the change in the state estimate. Indeed, when the measurements are linear, the Jacobian of the measurement function, H, is by definition constant. When the measurement function is nonlinear, this Jacobian varies with the state estimate. Each line of H corresponding to a measurement direction, this means that the measurement direction are changing from one step to another, because of the change in the state estimate.
3. As a result, the second update step mixes the very low uncertainty in the accurate direction with the high uncertainty in the coarse direction. Because of this "spillover", the uncertainty is too much reduced in the coarse direction.
4. Finally, the filter relies too much on its current estimate, and not enough on the new measurements. In this scenario, the filter does not converge.

6.2.2 Walk through an Example

In this part, we consider an example, and show step by step how the instability raises for the reasons mentioned above. The system considered is a vehicle lying in a plane (no motion), with range and bearing sensors located at the origin of the frame. The state consists of the abscissa and the ordinate of the vehicle position. The range sensor (accurate sensor) has an accuracy of $5 \cdot 10^{-3}$ [1]. The bearing sensor is coarse, with an accuracy of $\frac{\pi}{40}$ radian. The measurement uncertainty matrix is then given by

$$R = \begin{bmatrix} 2.5 \cdot 10^{-5} & 0 \\ 0 & 6 \cdot 10^{-3} \end{bmatrix}$$

The initial covariance matrix is given by

$$P = \begin{bmatrix} 100^2 & 0 \\ 0 & 100^2 \end{bmatrix}$$

so that the $1 - \sigma$ confidence area is a circle of radius 100. The initial position of the vehicle is set to

$$\mathbf{x} = \begin{bmatrix} 100 \\ 100 \end{bmatrix}$$

while the initial estimate is set to[2]

$$\hat{\mathbf{x}} = \begin{bmatrix} 20 \\ 80 \end{bmatrix}$$

which is in the confidence area given by the covariance matrix (the distance between the actual state $\mathbf{x}$ and the initial estimate $\hat{\mathbf{x}}$, or initial error, is 82).

Finally, the measurement matrix is given by the derivation of the range and bear-

[1] The accuracies are chosen not too different from each other (their ratio is 100), but still enough to show the instability raising

[2] The effect of the initial error on the final bias is represented on Fig. 6-7

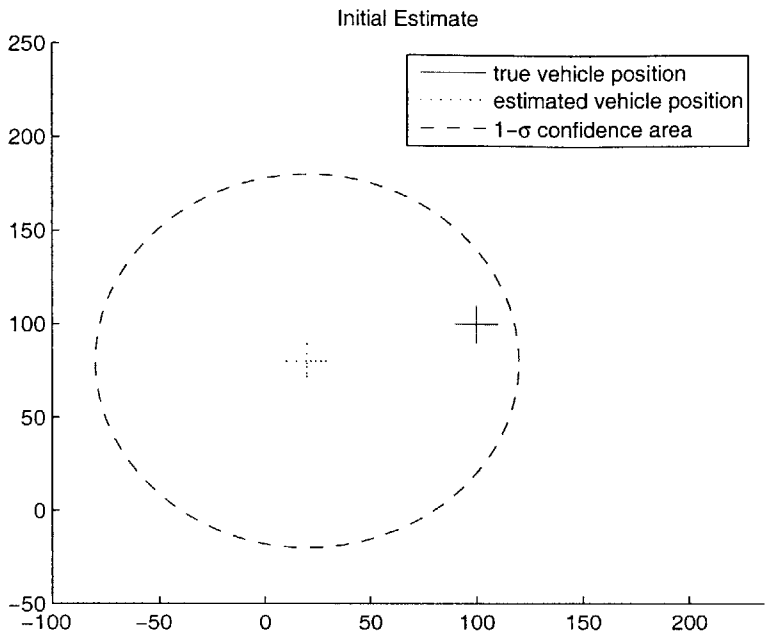

Figure 6-2: Initial Estimation

ing functions:

$$H = \begin{bmatrix} \frac{\hat{x}_1}{r} & \frac{\hat{x}_2}{r} \\ -\frac{\hat{x}_2}{r^2} & \frac{\hat{x}_1}{r^2} \end{bmatrix}$$

where

$$r = \sqrt{\hat{\mathbf{x}}_1^2 + \hat{\mathbf{x}}_2^2}$$

is the distance from the origin of the frame to the estimated position (range measurement). The initial situation is depicted on Fig. 6-2. The actual position of the vehicle lies within the uncertainty area (the circle of radius 100 centered at the estimated position).

We follow now the first two steps of the estimation process. In the first update step, the measurement matrix is

$$H \approx \begin{bmatrix} 0.2425 & 0.9701 \\ -1.2 \cdot 10^{-2} & 2.9 \cdot 10^{-3} \end{bmatrix}$$

The covariance matrix after the update is

$$P = \begin{bmatrix} 27.8123 & -6.9531 \\ -6.9531 & 1.7383 \end{bmatrix}$$

with eigenvalues $\lambda_1 \approx 2.5 \cdot 10^{-5}$ and $\lambda_2 \approx 29.6$.

As stated in the first section, P has become ill-conditioned in one step – the ratio of its eigenvalues is huge. Moreover, these eigenvalues corresponds respectively to the eigenvectors

$$\mathbf{e}_1 = \begin{bmatrix} 0.24 \\ 0.97 \end{bmatrix} \text{ and } \mathbf{e}_2 = \begin{bmatrix} -0.97 \\ 0.24 \end{bmatrix}$$

where $\mathbf{e}_1$ is in the radial direction and $\mathbf{e}_2$ is orthogonal to $\mathbf{e}_1$. These results show that the filter has a very small uncertainty in the range direction, and very large uncertainty in the bearing direction, which is the expected result because the range sensor is very accurate and the bearing sensor is coarse. The new position estimate is

$$\hat{\mathbf{x}} = \begin{bmatrix} 55.6 \\ 131.9 \end{bmatrix}$$

which has an associated range measurement of $\sqrt{55.6^2 + 131.9^2} = 143.1$. This value is close to the actual range to the vehicle, which is $\sqrt{100^2 + 100^2} = 141$. The bearing of the estimated position, $\arctan(\frac{131.9}{55.6}) \approx 1.17$ is far from the actual value, $\arctan(\frac{100}{100}) = \frac{\pi}{4} \approx 0.79$ which is reflected in the high eigenvalue of the covariance matrix.

This first Update Step is represented on Fig. 6-3. This figure shows that the high eigenvalue of the covariance in the direction of $\mathbf{e}_2$ takes the large bearing uncertainty into account. However, after the update, the state estimate has changed. Thus, the Jacobian of the measurement function, H, will be modified. In particular, the lines of H, which correspond to the measurements directions, will change: the directions of the coarse and accurate measurements will rotate. Then, the "spillover" mentioned earlier will make the uncertainty in the coarse direction too small.

Indeed, during the second update step (see Fig. 6-4), changes in the state estimate

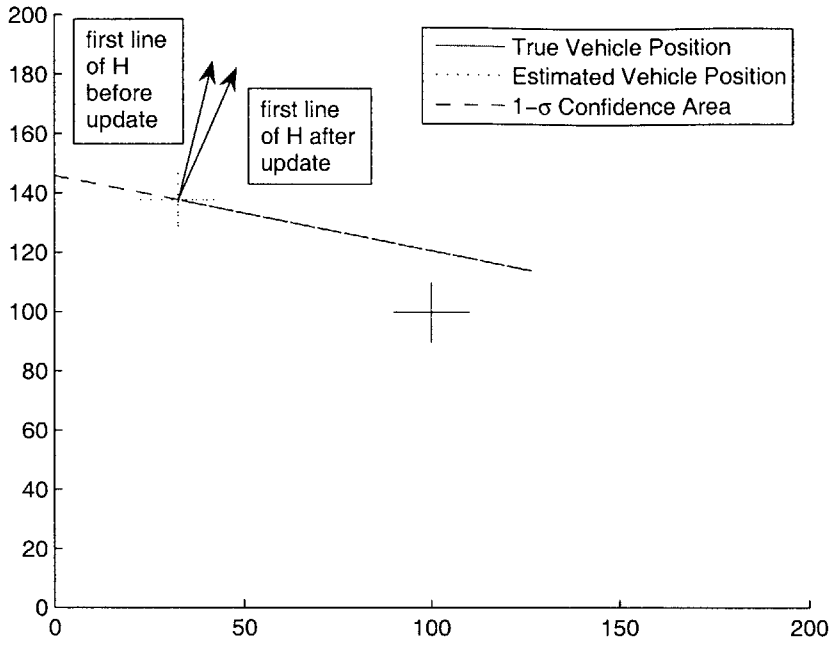

Figure 6-3: Estimation after First Update

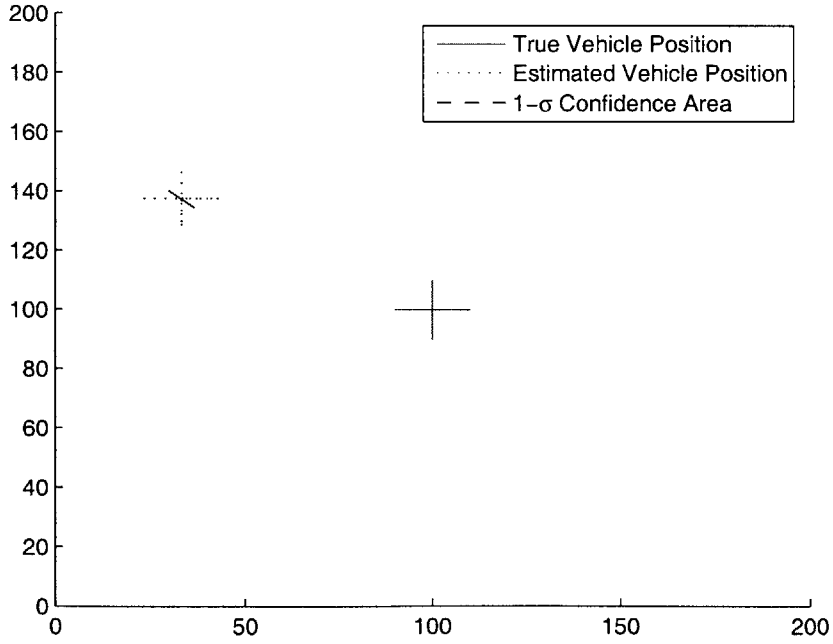

Figure 6-4: Estimation after Second Update

change the measurement matrix[3]

$$H = \begin{bmatrix} 0.3883 & 0.9215 \\ -6.4 \cdot 10^{-3} & 2.7 \cdot 10^{-3} \end{bmatrix}$$

The new state estimate is:

$$\hat{\mathbf{x}} \approx \begin{bmatrix} 44.9 \\ 134.5 \end{bmatrix}$$

so that $\|\mathbf{x} - \hat{\mathbf{x}}\| = 65$. This large value is almost entirely due to the error in the bearing estimation (the range of the estimated position is now 141.8, almost equal to the one of the actual position). The updated covariance matrix is

$$P \approx \begin{bmatrix} 1.9 \cdot 10^{-3} & -6 \cdot 10^{-4} \\ -6 \cdot 10^{-4} & 2 \cdot 10^{-4} \end{bmatrix}$$

with eigenvalues $\lambda_1 \approx 2.1 \cdot 10^{-3}$ and $\lambda_2 \approx 1.3 \cdot 10^{-5}$. The maximum eigenvalue of the covariance matrix is then 30000 times smaller than the actual error in the state estimate. On Fig. 6-4, the estimate is depicted after the second Update Step. The area of confidence is a small zone around the estimated position.

As a result, the covariance being small in comparison with the coarse measurements, the information contained in these measurements will not be included into the update. Fig. 6-5 represents the evolution of the error in the estimate. As expected, the error does not go to zero. Since the covariance was underestimated, the filter is not able to use the subsequent measurements to improve the estimate: the error remains very large until the end of the simulation.

A statistical simulation was performed. 10000 initial estimates were generated, according to the Gaussian distribution of covariance 100, centered at the true position

[3]Astonishingly, it is the very slight change in the first line of H–corresponding to the accurate range measurement–, not the large change in the second line, that is responsible for the instability, as it was shown through simulations, and in section 6.2.3

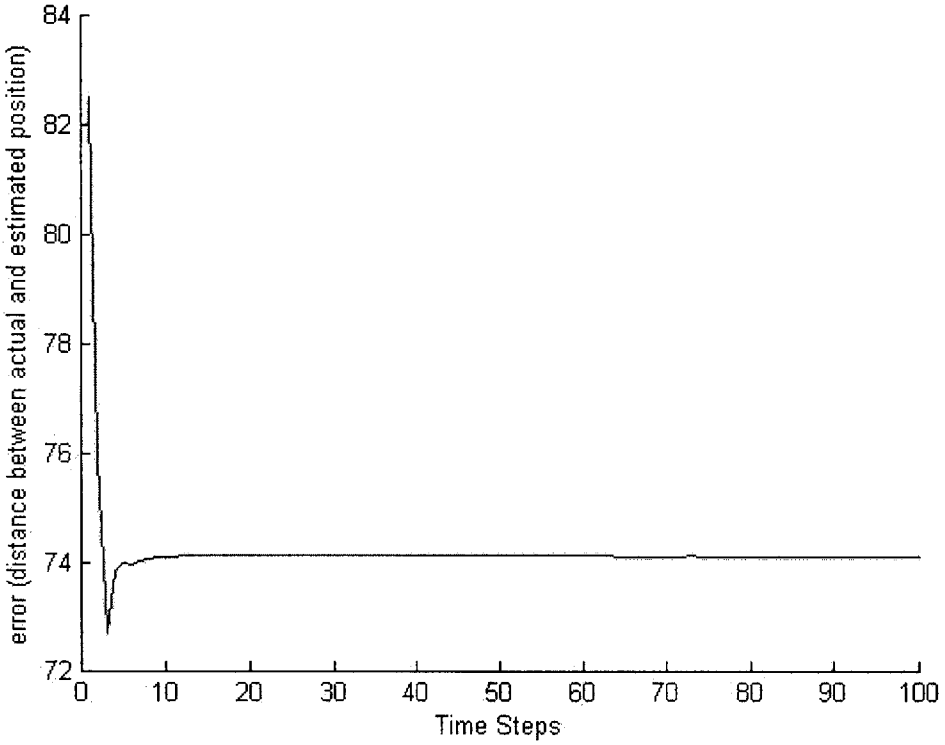

Figure 6-5: Evolution of the error between the actual state and the estimate for 100 steps

of the vehicle $\mathbf{x} = \begin{bmatrix} 100 & 100 \end{bmatrix}^T$. The position of the estimate after 100 steps was plotted for these initial values. Fig. 6-6 is a scatter plot of the estimates after 100 steps. Out of 10000, only 3929 (39.29%) were inside the confidence area. Probability theory states that 68.27% of the estimates should be inside the confidence area.

Fig. 6-7 represents the final bias in the estimate (after 100 steps) as a function of the initial error (between 0 and 100). Each value is an average on 10 random initial conditions with the given distance to the true vehicle position. The result is the perfect equality between the initial error and the final bias. This can be interpreted as the inefficiency of the EKF in this case: the EKF cannot correct the initial error to improve the estimate along the course of the algorithm.

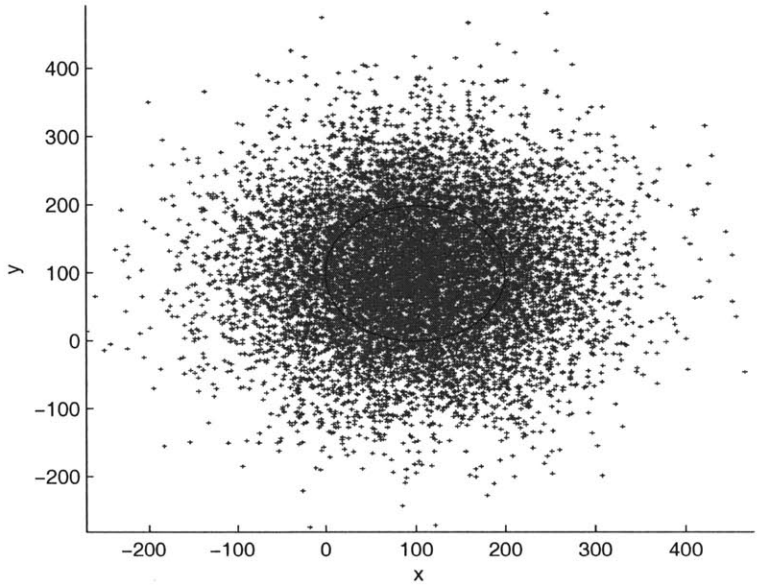

Figure 6-6: Repartition of the estimates after 100 steps for 10000 initial estimates. The circle represents the 1-σ area of confidence

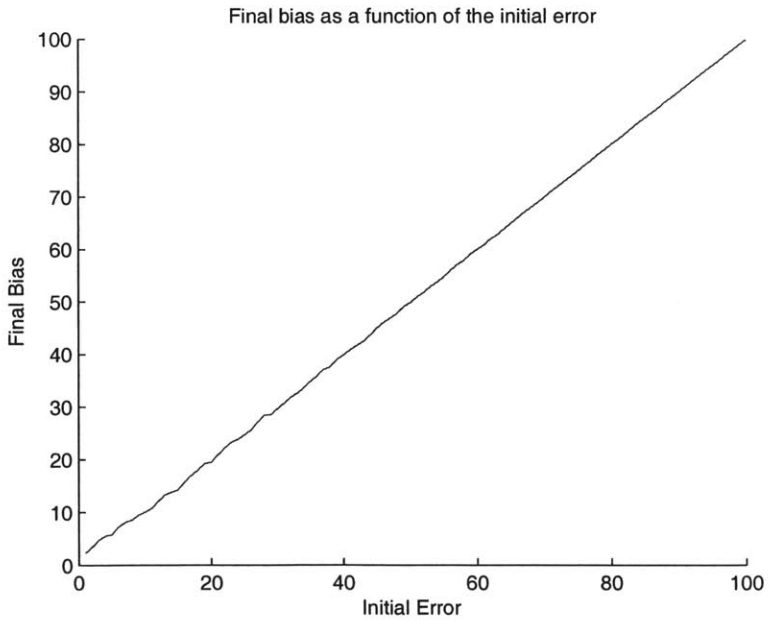

Figure 6-7: Effect of the initial error on the final bias

6.2.3 Detailed Analysis

In this part a more general case is analyzed, to show that the phenomenon mentioned above–the modification in the computation of H between the first and the following steps, that results in an over-reduction of the uncertainty in the coarse direction– is very likely to happen, and when it happens, is very likely to prevent the EKF from converging to the actual state of the system.

As in the previous example, the state considered here has two dimensions, with very different measurement accuracies. There is no motion. The measurement covariance is assumed to be

$$R = \begin{bmatrix} A & 0 \\ 0 & \epsilon \end{bmatrix}$$

where $A \gg \epsilon$. At the beginning of the simulation the covariance matrix is given by

$$P = P_0 = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$$

with $k \gg A$ and $k \gg \epsilon$. This analysis falls into three parts:

1. Show that the transpose of the second line of the measurement matrix becomes eigenvector of the covariance matrix after the first measurement update step.
2. Show that the covariance matrix becomes very ill-conditioned (very large ratio of eigenvalues) after the first measurement update step.
3. Show that, after the second measurement update, the actual state can be very far from the area of confidence (defined by the estimate and the covariance matrix).

The transpose of the second line of H_0 becomes eigenvector of P_0^+

To validate this conjecture, let us first show that a change of base can be performed, so that it can be assumed that

$$H_0 = \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix}$$

If

$$H_0 = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

an orthogonal matrix

$$O = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

is such that

$$H_0 O^T = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \tag{6.2}$$

as long as $\gamma\cos(\theta) + \delta\sin(theta) = 0$. This is obtained with $\theta = \arctan\left(-\frac{\gamma}{\delta}\right)$ if $\delta \neq 0$, and $\theta = \frac{\pi}{2}$ if $\delta = 0$. Thus, an orthogonal base change makes H_0 of the shape of Equation 6.2. It remains to show that the updated covariance matrix considered in the new base is the update of the covariance matrix in the new base. The following equalities hold

$$\begin{aligned} OP_0^+O^T &= O(I - K_0H_0)P_0O^T = OP_0O^T - OP_0H_0^T(H_0P_0H_0^T + R)^{-1}H_0P_0O^T \\ &= OP_0O^T - OP_0O^TOH_0^T\left(H_0O^TOP_0O^TOH_0^T + R\right)^{-1}H_0O^TOP_0O^T \\ &= P_0 - P_0\tilde{H}_0^T\left(\tilde{H}_0P_0\tilde{H}_0^T + R\right)^{-1}\tilde{H}_0P_0 \end{aligned}$$

where $\tilde{H}_0 = H_0O^T$ is the measurement matrix H in the changed base. The identity $O^TO = I$ was used, since O is an orthogonal matrix. The equality $OP_0O^T = P_0$ holds when P_0 is assumed to be identity times a scalar. These equalities show that the updated covariance matrix, in the new base, is the update of the initial covariance in the new base. Finally, if Oh_0 is an eigenvector of $OP_0^+O^T$, then $\left(OP_0^+O^T\right)Oh_0 = \alpha Oh_0$, and $P_0^+h_0 = \alpha h_0$, so that in this case h_0 is shown to be an eigenvector of P_0^+. This proves that a change of base is valid, so that H can be assumed to be of the shape $H_0 = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix}$. Now, if H_0 is divided by c and R by c^2, it suffices to prove the result for $c = 1$ (by dividing R by c^2, the assumption $A \gg \epsilon$ still holds). Thus, H is

assumed to be

$$H_0 = \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix}$$

The goal now is to prove that the transpose of the second line of H_0, which is $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, is an eigenvector of P_0^+. This is true if there exists α such that $P_0^+ h_0 = \alpha h_0$. Now, with $h_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, the condition becomes $\begin{bmatrix} P_{12}^+ \\ P_{22}^+ \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha \end{bmatrix}$, that is to say $P_{12}^+ \approx 0$. Now

$$|P_{12}^+| = | - \frac{k^2 ab\epsilon}{k^2a^2 + ka^2\epsilon + kb^2\epsilon + Ak + A\epsilon}|$$

Now $k^2a^2 + ka^2\epsilon + kb^2\epsilon + Ak + A\epsilon \geq k^2a^2$ implies that $|P_{12}^+| \leq \frac{b}{a}\epsilon$. If $b \gg a$, it means that the two lines of H are almost proportional, which means that the two measurements are almost parallel, making the system unobservable. If a is of the order of b or larger, then the last inequality shows that $|P_{12}^+| \ll 1$.

Thus, it was shown that after the first update step, the transpose of the second line of H_0 is eigenvector of the updated covariance matrix P_0^+. At the cost of a new base change, let us assume that P_0^+ is diagonal, and that the transpose of the first line of H_0 is also eigenvector, corresponding to the large eigenvalue of P_0^+ (which is true, for instance, for a range and bearing system).

The covariance matrix P_0^+ after the first update step is very ill-conditioned

Considering again that $H_0 = \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix}$, the two eigenvalues of P_0^+ can be computed. With the assumptions that

$$A \gg \epsilon, \quad a^2 + b^2 \ll \frac{A}{\epsilon},$$

the two eigenvalues of P_0^+ are given by

$$\lambda^+ = \frac{Ak}{A + ka^2},$$

$$\lambda^- = \frac{2\left(ka^2 + A\right)\epsilon}{A + ka^2}.$$

Their ratio is then

$$\frac{\lambda^-}{\lambda^+} = 2\left(\frac{a^2}{A} + \frac{1}{k}\right)\epsilon \ll 1,$$

from the assumptions. Thus, the first update step makes the covariance matrix P_0^+ being ill-conditioned, with a huge ratio of eigenvalues. For simplicity, $P_0^+ = P_1 = R$ is assumed.

After the second measurement update, the actual state can be very far from the area of confidence

In the preceding sections, it was shown that

- The transpose of the second line of H_0 is eigenvector of $P_0^+ = P_1$. P_1 is assumed diagonal for simplicity. Before the first step, it is assumed that, $H_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Then, after the first update step, because of the nonlinearity, H is changed. It is assumed that after the update, $H_1 = \begin{bmatrix} 1 & 0 \\ \delta & 1 \end{bmatrix}$ (H has been modified because of the change in the state estimate.

- P^+ is ill-conditioned, that is: the ratio of its eigenvalues is huge. $P_1 = R$ is assumed for simplicity.

The eigenvalues of the covariance matrix P_1^+ can be inferred, and it can be shown, under the assumptions $A \gg \epsilon$ and $\delta \gg \sqrt{\epsilon/A}$[4], that the greater eigenvalue of P_1^+ is $\lambda \approx 2\frac{\epsilon}{\delta^2}$. It will be shown here that the error after update, $\|x - \hat{x}^+\|$ can be much larger than λ. This can prevent the algorithm from finding an accurate state estimate.

[4]Note that this threshold is very low, from the assumption $A \gg \epsilon$, so that it is very likely that, because of the first update and the non linearity, a larger modification occurs in H, preventing convergence as we show here

The Kalman gain is computed through:

$$K_1 = P_1 H_1^T \left(H_1 P_1 H_1^T + R \right)^{-1} = \begin{bmatrix} \frac{2\epsilon}{A\delta^2 + 4\epsilon} & \frac{A\delta}{A\delta^2 + 4\epsilon} \\ -\frac{\epsilon\delta}{A\delta^2 + 4\epsilon} & \frac{2\epsilon}{A\delta^2 + 4\epsilon} \end{bmatrix}$$

If the state update equation is considered

$$\hat{x}_1^+ = \hat{x}_1^- + K_1 \left(y_1 - h(\hat{x}_1^-) \right)$$

Here, y_1 is the measurement, which is a function of the actual state x, through: $y_1 = h(x)$. This equation can be written as an update on the error before, $e^- = x - \hat{x}^-$, and after, $e^+ = x - \hat{x}^+$, through

$$\begin{bmatrix} e_1^+ \\ e_2^+ \end{bmatrix} = \begin{bmatrix} e_1^- - \frac{2\epsilon\tilde{y}_1 - A\delta\tilde{y}_2}{A\delta^2 + 4\epsilon} \\ e_2^- - \frac{\epsilon\delta\tilde{y}_1 + 2\epsilon\tilde{y}_2}{A\delta^2 + 4\epsilon} \end{bmatrix}, \tag{6.3}$$

where $\tilde{y}$ denotes the discrepancy between the measurement and the predicted measurement

$$\tilde{y} = h(x) - h(\hat{x})$$

From Equation 6.3, it can be inferred that the error after update can be much larger than the largest eigenvalue of P_1^+. For instance, if $e_2^- = \tilde{y}_2 = 0$ and $e_1^- = \tilde{y}_1 \approx A$, then the error after update is given by

$$e_1^+ = \frac{A\delta^2 + 2\epsilon}{A\delta^2 + 4\epsilon} e_1^- \approx A \gg 2\frac{\epsilon}{\delta^2} = \lambda_{max}$$

Thus, the problem depicted on Fig. 6-4 (the error after update can be much larger than the largest eigenvalue of P_1^+) can happen again.

Conclusion

As shown in this section, the scenario by which the Extended Kalman Filter does not estimate properly the state of the system is the following

1. During the first update step, the covariance matrix is shrunk in the direction of the accurate measurement.

2. During the second update step, the direction of the accurate measurement is often modified because of the nonlinearity and the change in the state estimate.

3. As a result, the second update step mixes the very low uncertainty in the accurate direction with the high uncertainty in the coarse direction, causing an over-reduction of the uncertainty in the coarse direction.

4. Finally, the filter trusts too much its current estimate, and does not use enough the subsequent measurements.

To fix this problem, a method based on "Bumping-Up" the measurement uncertainty matrix R is proposed.

6.3 A Method to fix the Instability: Bumping-Up the Measurement Covariance Matrix

6.3.1 Principle of the Bump-Up R Method

The reason why the EKF does not converge is the over-reduction of the covariance matrix in a direction. The method presented here is based on increasing artificially the covariance of the measurements. The goal of this method is that *the estimate becomes accurate before P becomes ill-conditioned.* More precisely, the goal is to reduce the two eigenvalues of P at the same speed, at least at the beginning. Then, when the eigenvalue corresponding to the coarser sensor reaches this sensor's accuracy, the other eigenvalue will keep on decreasing. P will then also be ill-conditioned, but at a time when the estimation is already quite accurate, so that the error in H will be much smaller. As a result, the undesired effect described in the previous section, leading to an over reduction of the larger eigenvalue of P will not happen. The idea is

to make the accuracy of the sensors high when the uncertainty of the estimate is high, and low when the uncertainty of the estimate is low. In other words, the accuracy of the sensors will follow the evolution of the covariance of the estimate. This is why R is bumped-up by HPH^T.

6.3.2 Bump-Up R Method: Presentation and Analytical Analysis

The method consists of replacing R by $\hat{R} = R + HPH^T$. The updated covariance matrix becomes

$$P^+_{bump-up} = \begin{bmatrix} \frac{(6\epsilon + A\delta^2)\,A}{9\epsilon + 2\,A\delta^2} & -\frac{\epsilon A\delta}{9\epsilon + 2\,A\delta^2} \\ -\frac{\epsilon A\delta}{9\epsilon + 2\,A\delta^2} & 2\frac{(3\epsilon + A\delta^2)\,\epsilon}{9\epsilon + 2\,A\delta^2} \end{bmatrix} \tag{6.4}$$

Equation 6.4 shows that:

- If $\delta = 0$ (no error in H), then the covariance matrix reduces to

$$P^+_{bump-up} = \begin{bmatrix} \frac{2}{3}A & 0 \\ 0 & \frac{2}{3}\epsilon \end{bmatrix}$$

  The covariances in the two directions are reduced less than they would with no bump-up (2/3 instead of 1/2). Thus, the filter will be slower, but will eventually converge.

- If $A\delta^2 \gg \epsilon$, or equivalently $\delta \gg \sqrt{\frac{\epsilon}{A}}$ (large change in H matrix, which is the cause of instability in the EKF with no Bump-Up), then the updated covariance would become

$$P^+_{bump-up} = \begin{bmatrix} \frac{1}{2}A & 0 \\ 0 & \epsilon \end{bmatrix}$$

There is no over-reduction of the coarse direction covariance, and the filter will not experience the divergence issues.

6.3.3 Simulation Results

The Bump-Up Method was implemented for a single vehicle, moving in a plane. As in the above example, a range and bearing sensor system is located at the origin of the frame. An Extended Kalman Filter is implemented, and compared to its modification through the Bump-Up R method. The parameters used in this simulation are summarized in Table. 6.1.

Table 6.1: Parameters for the Bump-Up R Simulation

variable definition	variable value
initial position uncertainty	10
initial velocity uncertainty	0.1
initial position range	[50,100]
initial velocity range	[-1,1]

For the measurement covariance matrix

$$R = \begin{bmatrix} 0.001^2 & 0 \\ 0 & \left(\frac{\pi}{40}\right)^2 \end{bmatrix},$$

the result is presented on Fig. 6-8 and 6-9. The estimation performed with the standard EKF diverges (the error remains above 15 after 100 steps). With the BUP method, the estimate converges.

Statistics on the convergence of the two filters are presented on Fig 6-10. The bearing accuracy was set to $\frac{\pi}{40}$. The probability to converge is plotted, computed on ten runs for each value of the range accuracy. The algorithm was said to converge when after 100 time steps the error (distance between the true position of the vehicle and the estimate) was less than 1. The x axis represents the range accuracy in a logarithmic scale. The red curve represents the statistics for the Extended Kalman

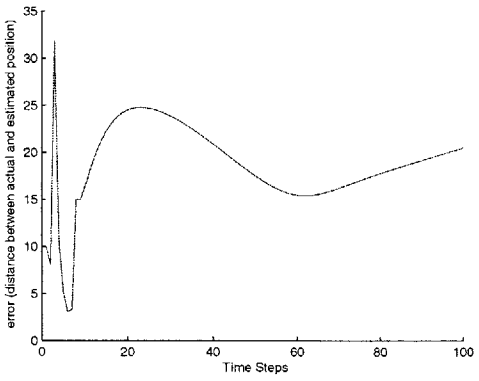

Figure 6-8: The error does not go to zero with the Extended Kalman Filter

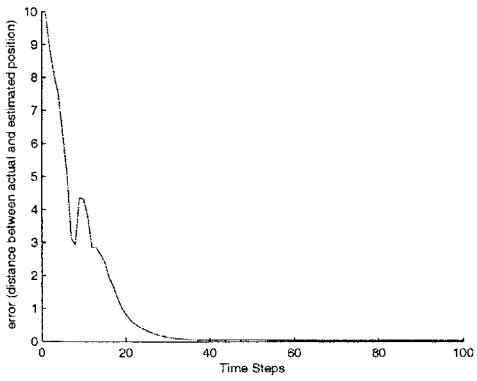

Figure 6-9: With the Bump-Up R Method, the estimation converges

Filter, the green one, for the Bump-Up R Method. The BUP is shown to obtain much better performances, especially when the ratio between the accuracies is very large. When the range accuracy is 10000 times greater than the bearing accuracy, the EKF almost never converge, whereas the EKF BUP converges in 80% of the cases.

6.4 Conclusion

The reason for the instability of the Extended Kalman Filter was identified. The linearization of the measurement function around the estimate is responsible for a modification of the measurement directions between the first and the second measurement update steps. This modification of direction leads to "spillover" between the large and small eigenvalues of the covariance matrix. As a result, the direction of coarse measurement corresponds to a low eigenvalue. Because of this over reduction of the covariance in the coarse direction, the filter does not use the subsequent measurements enough. Finally, it fails in improving the estimate over time.

The Bump-Up R method was shown to be an efficient method to fix this issue. By increasing artificially the covariance of the measurements, the filter is forced to decrease gradually the covariance in both directions. As a result, the spillover is avoided at the cost of a reduction in the convergence speed of the algorithm.

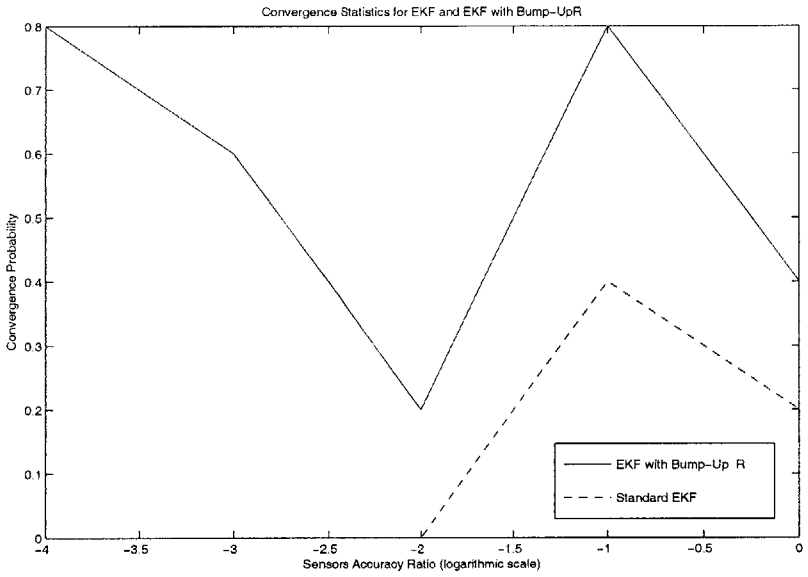

Figure 6-10: Comparison between EKF and EKF BUP: the latter is much more likely to converge, especially when the accuracies ratio is huge

Chapter 7

Conclusion

7.1 Architectures Comparisons

This thesis analyzed and compared various algorithms. The criteria used for these comparisons were

- The accuracy reached by the estimate;
- The computational complexity required;
- The synchronization degree required;
- The communication load necessary.

The Centralized Algorithm was shown to be optimal, but necessitating huge computations to be carried out. As a result, the delay introduced in the computations degrade the real-time performance of this algorithm, so that its performances in real-time got worse as the number of spacecrafts in the fleet increases.

The Decentralized Scheme was shown to be very efficient. In distributing the computation across the fleet, it allowed low computational complexity. While the computations performed were slightly approximate with respect to the one performed in the Centralized Algorithm, the low computational complexity allows fast estimation

refreshment, so that the real-time performance is good and scales very well as the number of satellites increases. The synchronization and communication requirements were emphasized as the main flaws of the Decentralized Algorithm. Less synchronized versions of the Decentralized Scheme were shown to address this issue, but at the cost of a deteriorated accuracy.

The Hierarchic Scheme was found efficient in maintaining a good accuracy, while spreading the computational task across the clusters. The synchronization requirement was also maintained low thanks to the various levels of hierarchy. However, the communication requirement is the main flaw of these algorithms, for they necessitate heavy communication between the Sub-Clusters and the Super-Cluster.

7.2 Two Particular Problems

Two particular problems were studied in the course of this research.

A particular issue raised in the Magnetospheric Multiscale Mission was raised. Delayed measurements were received by the satellites of the fleet during the estimation process. Including these measurements appeared a challenge because the optimal process for doing so required large data to be saved, and huge computations to be carried out. Three methods to include the delayed measurements were analyzed and compared. One method was shown to be efficient in terms of accuracy, while requiring no data storage and low computational complexity.

A particular instability of the Extended Kalman Filter was analyzed. When two sensors of very different accuracies are used, the estimation performed by the filter often does not converge to the actual state of the system. The reason for this instability was explained. One possible method for fixing this instability was proposed and analyzed.

7.3 Future Work

The analysis performed throughout the course of this research were not definitive. Some algorithms, especially within the Hierarchic Scheme, were not analyzed as precisely as others. Moreover, the analytical studies were performed mostly on simplified models, so that they do not provide accurate information about the real systems. The numerical studies were performed with some parameters only, and could be developed as well for other scenarios. Such studies could be performed, and may lead to further improvements. Moreover, the performances of the Decentralized and Hierarchic Architectures were shown to be improvements with respect to some criteria, while they were oftentimes a deterioration with respect to other. Further research could lead to the design of innovative architectures able to improve or maintain the performance with respect to all criteria.

Appendix A

Appendix for Chapter 5

A.1 The Blending Phase of Approach 3

This appendix presents the derivation of the equations for the blending phase of Approach 3 (Equation 5.14). The first paragraphs presents a straight-forward approach that was used first, and its drawbacks. The second and third paragraphs analyze a the problem in general (in the 2-D in n-dimensional cases). Finally, the fourth paragraph derives the equations for the blending phase.

A.1.1 A Straight-Forward Approach and its Drawbacks

A straight-forward approach for the blending phase of Approach 3 would be as follows. From the STF, the considered satellite has the estimate $\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$ and the related covariance P on its local state and on the remote satellites state. The estimate for the local satellite is a result of the Schmidt-Kalman Filter, while the estimate on the remote satellite is a mere propagation of the estimate received from them. From the LTF, the satellite has also an estimate $\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}$ and its covariance $\tilde{P}$ for its local state, and the remote satellites state. These estimates were obtained by the prediction al-

gorithm described in the last paragraph. The blending can then be performed by considering $\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$ and $\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}$ as measurements of the global state of the fleet, with covariances P and $\tilde{P}$. The two estimate would then be blended according to

$$\begin{aligned} \begin{bmatrix} \hat{x}^+ \\ \hat{y}^+ \end{bmatrix} &= \left(\tilde{P} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} + P \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} \right) (P + \tilde{P})^{-1} \\ P^+ &= (P\tilde{P})(P + \tilde{P})^{-1} \end{aligned} \tag{A.1}$$

However, this approach has two drawbacks

1. This blending double counts some information. Indeed, for the local satellite, the information included in the propagated state through the slow period is already included in the estimate from the STF. In the same fashion, the estimates for the remote satellite from the STF -mere propagation- is included in the fast estimate;
2. This blending is very computationally intensive because of the inversion of a square matrix of size $6N \times 6N$ $(P + \tilde{P})$.

Thus, this blending process was improved, by both avoiding double counting and reducing the complexity. To derive the equations for this blending process, the following problem needs to be solved: given an estimate on two variables x and y, and the related covariance, how should the cross-correlations between x and y be updated when the estimate on y is replaced by a new one ?

A.1.2 2-D Case

Let us first consider two one-dimensional variables x and y. Let us assume that an estimate is available for them, together with a covariance matrix

$$P = \begin{bmatrix} P_{xx} & P_{xy} \\ P_{yx} & P_{yy} \end{bmatrix}$$

Now, let us assume that a better estimate on y is provided. Let us assume that the information included in the current estimate is also included in this new estimate, so that the resulting covariance for y is $\tilde{P}_{yy}$ (since there would be double counting in considering the former and the new estimate as independent estimates). Now, to know how to change the other terms of the covariance matrix and how to update $\hat{x}$, *this update will be considered as a measurement on* y. The covariance R of this measurement will then be computed so that the new covariance of y is $\bar{P}_{yy}$. The updated covariance is (Kalman filter)

$$P^+ = \left(I - PH^T\left(HPH^T + R\right)^{-1} H\right) P$$

$$P^+ = \begin{bmatrix} P_{xx} - \frac{P_{xy}^2}{P_{yy}+R} & \frac{RP_{xy}}{P_{yy}+R} \\ \frac{RP_{xy}}{P_{yy}+R} & \frac{RP_{yy}}{P_{yy}+R} \end{bmatrix}$$

Now, R has to be chosen so that

$$\frac{RP_{xy}}{P_{yy}+R} = \tilde{P}_{yy} => R = \frac{P_{yy}\tilde{P}_{yy}}{P_{yy} - \tilde{P}_{yy}}$$

Then the covariance becomes, in terms of $\tilde{P}_{yy}$ instead of R

$$P^+ = \begin{bmatrix} P_{xx} - \frac{P_{xy}^2}{P_{yy}}\frac{P_{yy}-\tilde{P}_{yy}}{P_{yy}} & P_{xy}\frac{\tilde{P}_{yy}}{P_{yy}} \\ P_{xy}\frac{\tilde{P}_{yy}}{P_{yy}} & \tilde{P}_{yy} \end{bmatrix}, \tag{A.2}$$

The state update is

$$\begin{bmatrix} x \\ y \end{bmatrix}^+ = \begin{bmatrix} x^- + \frac{Pxy(P_{yy}-\tilde{P}_{yy})}{P_{yy}^2}(\tilde{y} - y^-) \\ \tilde{y} \end{bmatrix}, \tag{A.3}$$

Where the kalman filter update for y has been replaced by the simple replacement of y^- by $\tilde{y}$.

One needs to check that the resulting matrix is a covariance matrix

- every term is positive: this is obvious for $\tilde{P}_{yy}$ and $\tilde{P}_{xy}$. Then

$$P_{xx}^{+} = P_{xx} - \frac{P_{xy}^2}{P_{yy}} \frac{P_{yy} - \tilde{P}_{yy}}{P_{yy}} = \frac{1}{P_{yy}} \left(P_{xx} P_{yy} - P_{xy}^2 \left(\frac{P_{yy} - \tilde{P}_{yy}}{P_{yy}} \right) \right)$$

  so that

$$P_{xx}^{+} > \frac{1}{P_{yy}} \left(P_{xx} P_{yy} - P_{xy}^2 \right)$$

  .

- $P_{xx}^{+} P_{yy}^{+} - \left(P_{xy}^{+} \right)^2 > 0$. One can write $P_{xx}^{+} P_{yy}^{+} - \left(P_{xy}^{+} \right)^2 = \frac{\tilde{P}_{yy}}{P_{yy}} \left(P_{xx} P_{yy} - P_{xy}^2 \right)$, which is positive since P is a covariance matrix.

A.1.3 n-Dimensional Case

Let us walk through the same process for the n-dimensional case. The update on y is first considered as being a measurement of covariance R. The update is then

$$P^{+} = \begin{bmatrix} P_{xx} - P_{xy} \left(P_{yy} + R \right)^{-1} P_{yx} & P_{xy} \left(P_{yy} + R \right)^{-1} R \\ R \left(P_{yy} + R \right)^{-1} P_{yx} & R \left(P_{yy} + R \right)^{-1} P_{yy} \end{bmatrix}$$

Then

$$R = \left(I - \tilde{P}_{yy} P_{yy}^{-1} \right)^{-1} \tilde{P}_{yy}$$

Thus

$$P^{+} = \begin{bmatrix} P_{xx} - P_{xy} \left(I - P_{yy}^{-1} \tilde{P}_{yy} \right) P_{yy}^{-1} P_{yx} & P_{xy} P_{yy}^{-1} \tilde{P}_{yy} \\ \tilde{P}_{yy} P_{yy}^{-1} P_{yx} & \tilde{P}_{yy} \end{bmatrix}, \tag{A.4}$$

The state update is

$$\begin{bmatrix} x \\ y \end{bmatrix}^{+} = \begin{bmatrix} x^{-} + P_{xy} (I - P_{yy}^{-1} \tilde{P}_{yy}) P_{yy}^{-1} (\tilde{y} - y^{-}) \\ \tilde{y} \end{bmatrix}, \tag{A.5}$$

When applied to the scalar case, Equations A.4 and A.5 give Equations A.2 and A.3 found in Section A.1.2.

A.1.4 Equations for the Blending Process

Based on the analysis of the previous paragraph, the new blending process performs the following operations

1. Start from the estimate computed by the STF.
2. Then, it is desired to include the estimate for the remote satellites from the long time-step filter (LTF), since it contains the most recent update on them. However, since this update consists of a new information about the remote satellites, it has an impact on the local satellite state estimate through the cross-correlations. This appendix has derived the state and covariance update to include the estimate of y from the long time-step filter to the estimate from the short time-step filter. Calling $\tilde{y}$ and $\tilde{P}_{yy}$ the estimate and covariance of the remote satellites from the long time-step filter, and $\hat{x}$, $\hat{y}$ and P the estimate and covariance from the STF, the new state estimate and related covariance matrix are given by Equation 5.14

$$\begin{aligned}
\begin{bmatrix} \hat{x}^+ \\ \hat{y}^+ \end{bmatrix} &= \begin{bmatrix} \hat{x} + P_{xy}\left(I - P_{yy}^{-1}\tilde{P}_{yy}\right)P_{yy}^{-1}(\tilde{y} - \hat{y}) \\ \tilde{y} \end{bmatrix} \\
P^+ &= \begin{bmatrix} P_{xx} - P_{xy}\left(I - P_{yy}^{-1}\tilde{P}_{yy}\right)P_{yy}^{-1}P_{yx} & P_{xy}P_{yy}^{-1}\tilde{P}_{yy} \\ \tilde{P}_{yy}P_{yy}^{-1}P_{yx} & \tilde{P}_{yy} \end{bmatrix}
\end{aligned}$$

This way of performing blending avoids double counting, since each estimate is copied from the filter that contains the most recent/accurate estimate on it. Moreover, the inversion of a matrix of size $6N \times 6N$ is replaced by the inversion of a block-diagonal matrix of size $6(N-1) \times 6(N-1)$ (P_{yy}).

A.2 Complexity Analysis

This appendix computes the complexity of the different algorithms implemented. The size of the state vector is $s = 6$.

A.2.1 Complexity Analysis for the Prediction Step

The complexity of the prediction step is given by

- State Time Update: s^2 to compute Φx, s to add the acceleration.
- Covariance Time Update: s^3 to perform $P\Phi^T$, s^3 to multiply by Φ. Finally, adding Q needs s^2 operations.

Thus, the complexity to compute this prediction step for one satellite is $2s^3 + 2s^2 + s = 510$. Since each satellite performs this computation for all satellites, the total complexity for this prediction step is $510N = 2040$.

A.2.2 Complexity Analysis for the Update Step

The complexity of an update step is given by

- Computation of α. $(sN)^2(N+2)$ to compute $\begin{bmatrix} H & J \end{bmatrix} P$, $(N+2)^2(sN)$ to multiply it by $\begin{bmatrix} H^T \\ J^T \end{bmatrix}$, and $(N+2)^2$ to add R.
- Kalman gain computation. The computations of $P_{xx}H^T$ and $P_{xy}J^T$ were already performed to compute α. The complexity to inverse α is $(N+2)^3$, and the complexity of multiplying $(P_{xx}H^T + P_{xy}J^T)$ by α^{-1} is $s(N+2)^2$.
- State measurement update. $(N+2)$ for $(z - \hat{z})$, $s(N+2)$ to multiply by K, s to add x^-.
- Covariance measurement update. $s^2(N+2)$ for KH, s to subtract to I, $s^2(N+2)(N-1)$ for KJ; P_{xx}: s^3 for $(I-KH)P_{xx}$, $s^3(N-1)$ for KJP_{yx}, s^2 to add these two; P_{xy}: $s^3(N-1)$ for $(I-KH)P_{xy}$, $s^3(N-1)^2$ for KJP_{yy}, $s^2(N-1)$ to add these two; P_{yx}: $s^2(N-1)$.

As a result, the total complexity for the update step is: $(sN)^2(N+2) + (sN)(N+2)^2 + (N+2)^3 + (s+1)(N+2)^2 + s^3(N-1)^2 + 36(N+2)(N-1) + (2s^3+2s^2)(N-1) + (s^2+s+1)(N+2) + s^3 + s^2 + 2s = 9414$ for $N = 4$.

A.2.3 Complexity Analysis for the Blending Phase

The complexity of the blending phase is

- State Update. $s^3(N-1)$ to compute P_{yy}^{-1}, $s^3(N-1)$ to compute $P_{yy}^{-1}\tilde{P}_{yy}$, $s(N-1)$ to subtract it to I, $s^3(N-1)$ to multiply by P_{yy}^{-1}, $s^3(N-1)^2$ to multiply by P_{xy}, $s(N-1)$ for $(\tilde{y}-\hat{y})$, and s to add it to $\hat{x}^-$.
- Covariance Update. $P_{xy}\left(I - P_{yy}^{-1}\tilde{P}_{yy}\right)P_{yy}^{-1}$ was already computed for the state update. Then: $s^3(N-1)$ to multiply by P_{yx}, s^2 to subtract it to P_{xx}, $s^3(N-1)^2$ to multiply $P_{yy}^{-1}\tilde{P}_{yy}$ to P_{xy}, $s^2(N-1)$ to take the transpose of P_{xy}^+.

Thus, the complexity of the blending phase is $2s^3(N-1)^2+4s^3(N-1)+s^2(N-1)+2s(N-1)+s^2+s=6666$.

A.3 Why the Gain of Using Accurate Crosslink-Range Measurements can be Low

The gain of using the crosslink range measurements is low (as seen on Fig. 5-5 in Chapter 5). The reason is that the accurate crosslink range measurements do not make the system observable by themselves alone. The system is only observable when adding the GPS measurements. To understand this, let us consider a 2-D system with two measurements: a coarse GPS measurement, and an accurate bearing measurement. From the GPS measurement only, the algorithm can compute a coarse estimate of the position of the vehicle (clear area on Fig. A-1). Now, if the accurate bearing measurement is added, the uncertainty is reduced (dark area on Fig. A-1).

However, the error in the estimation can still be large, since the bearing observation does not make by itself the system observable: there are still directions in which the estimation is coarse. A similar phenomenon happens in the MMS mission scenario: the reduction of uncertainty due to adding the range measurements is less than expected at first because these measurements by themselves do not make the

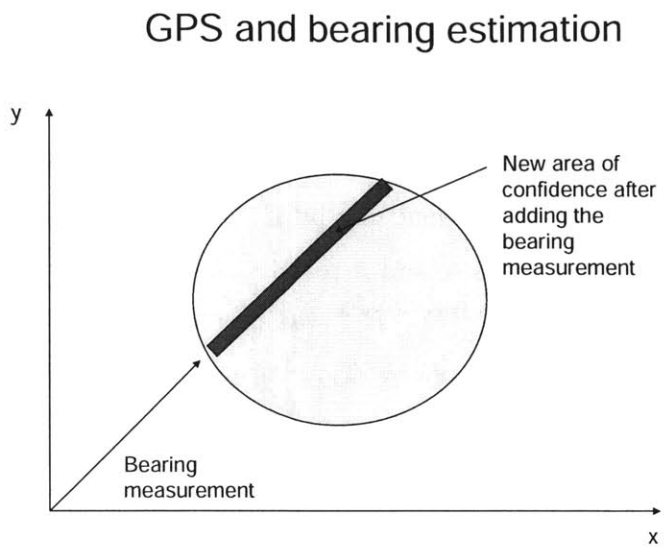

Figure A-1: GPS and bearing estimation: the error in the estimate even with the bearing measurement can still be large

system observable. This discussion is also related to the notion of Position Dilution of Precision (PDOP), mentioned in Chapter 2 (Section 2.3.3, [20]).

A.4 Gain of the Update From Other Satellites

This section corresponds to Section 5.4.1. Fig. A-2 represents the evolution of the covariance when an update is performed for the remote satellites. On the right side of this figure, the complexity for every time step is represented. Fig.A-3 represents the same evolution of the covariance when no update is performed for the remote satellites. Finally, Fig. A-4 is the difference between these two graphs, or the gain in terms of covariance of performing an update on the remote spacecrafts. As emphasized in Section 5.4.1, this gain is around 1m for the three approaches.

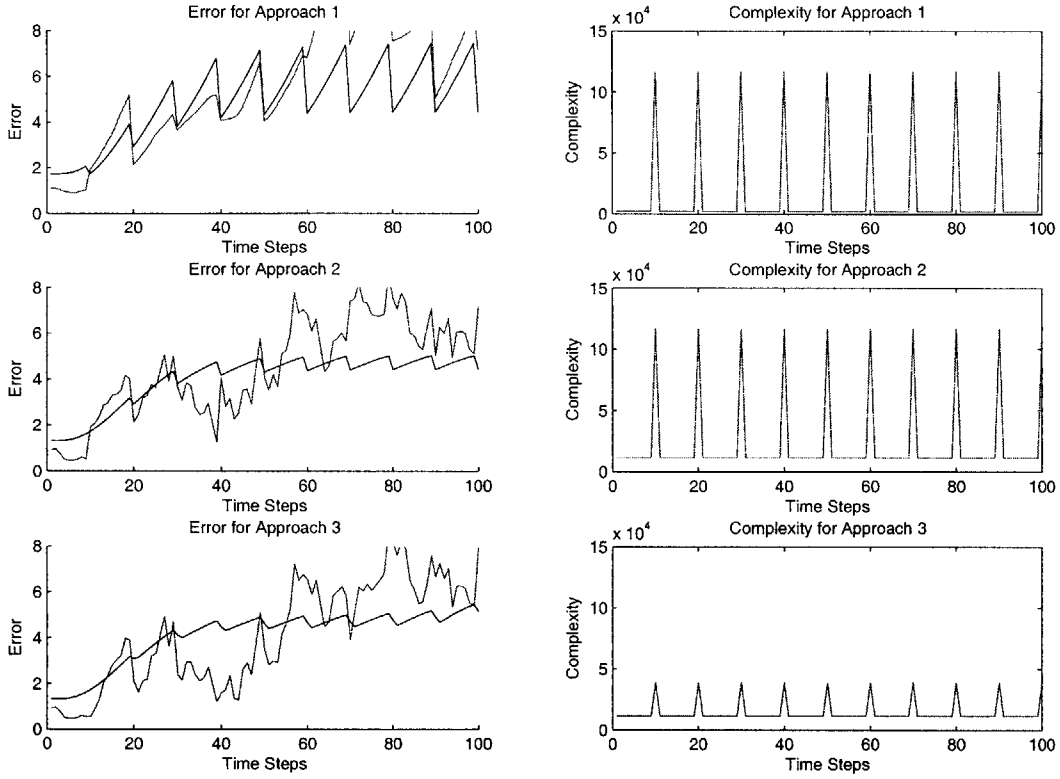

Figure A-2: Results with update for other satellites states and covariances

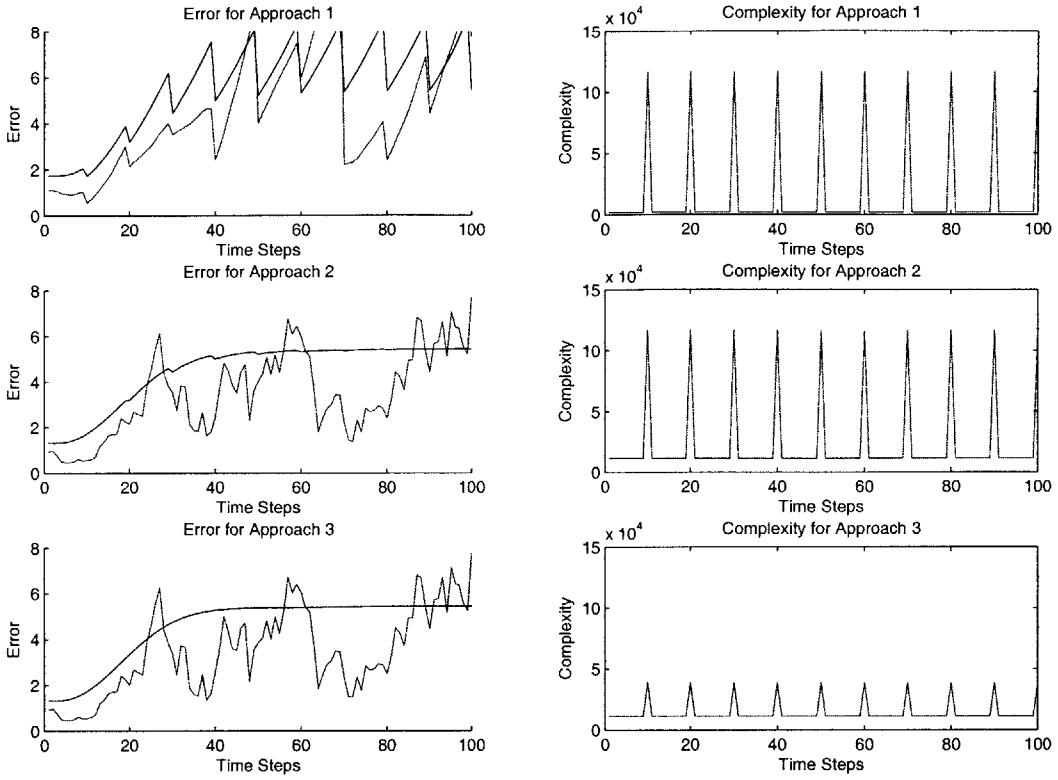

Figure A-3: Results without update for other satellites states and covariances

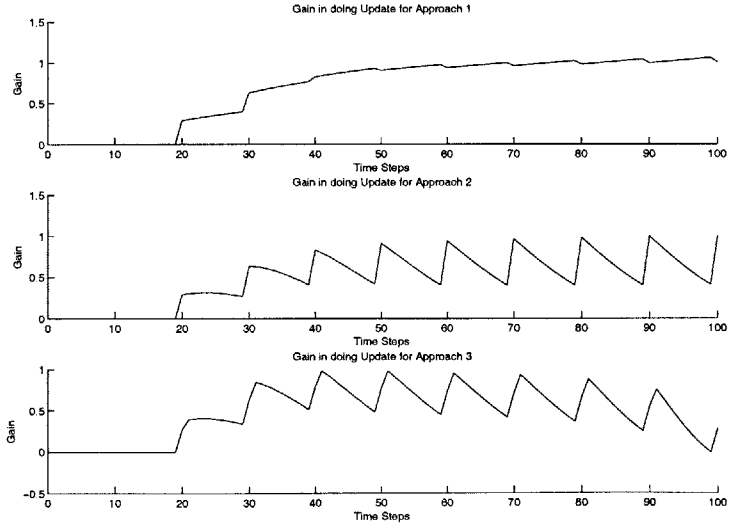

Figure A-4: Difference between accuracy with and without update

Bibliography

[1] F. H. Bauer, K. Hartman, J. P. How, J. Bristow, D. Weidow, and F. Busse, "Enabling Spacecraft Formation Flying through Spaceborne GPS and Enhanced Automation Technologies," *ION-GPS Conference*, Sept. 1999 (pp. 369–384).

[2] F. D. Busse, "Precise Formation-State Estimation in Low Earth Orbit using Carrier Differential GPS," Ph.D. thesis, Stanford University, Dept. of Aero/Astro, Nov. 2002.

[3] C. W. Park, "Precise Relative Navigation using Augmented CDGPS," Ph.D. thesis, Stanford University, Dept. of Mech. Eng., June 2001.

[4] P. Ferguson and J. P. How, "Decentralized Estimation Algorithms for Formation Flying Spacecraft," *Proceedings of the AIAA GNC Conf.*, Aug. 2003 (Paper 2003-5442).

[5] M. Mandic, L. Breger, and J. P. How, "Analysis of Decentralized Estimation Filters for Formation Flying Spacecraft," presented at the *AIAA GNC Conf.*, Aug. 2004 (AIAA-2004-5135).

[6] D. Kelbel, T. Lee, A. Long, J. Russell Carpenter and C. Gramling, "Evaluation of relative navigation algorithms for formation-flying satellites using GPS," *Proceedings of the 2001 Flight Mechanics Symposium*, NASA Goddard Space Flight Center, Greenbelt, Maryland, June 19-21, 2001.

[7] http://www.csc.com/aboutus/lef/mds67_off/uploads/CSCPapers2005_Relative Navigation.pdf, last accessed dec. 2005.

[8] J. R. Carpenter, C. Gramling et al., "Relative Navigation of Formation-Flying Satellites," *Proceedings of the International Symposium on Formation Flying*, Toulouse France, Oct. 2002.

[9] D. Kelbel, T. Lee, A. Long, J. Russell Carpenter and C. Gramling, "Relative Navigation Algorithms for Phase 1 of the MMS Formation," Flight Mechanics Symposium, 25-30 Oct. 2003, Greenbelt, MD, USA.

[10] M. S. Grewal and A. P. Andrews, Kalman Filtering, Theory and Practice using MATLAB, Prentice-Hall Inc., 1993.

[11] G. M. Siouris, An Engineering Approach to Optimal Control and Estimation Theory, John Wiley and Sons, 1996.

[12] A. Gelb, Applied Optimal Estimation, Massachusetts, MIT press, 1974.

[13] B. Southall, B. F. Buxton and J.A. Marchant, "Controllability and Observability: Tools for Kalman Filter Design," *British Machine Vision Conference*, 1998.

[14] Z. Chen, "Local Observability and Its Application to Multiple Measurement Estimation," *IEEE Trans. on Industrial Electronics*, 1991 (38(6):491-496).

[15] J. P. Hespanha, D. Liberzon, E.D. Sontag, "Nonlinear Observability and an Invariance Principle for Switched Systems," *Proc. IEEE Conf. Decision and Control, Las Vegas, IEEE Publications*, 2002 (pp. 4300-4305).

[16] S. Hong, M.H. Lee, H.-H. Chun, S.-H. Kwon and J.L. Speyer, "Observability of Error States in GPS/INS Integration," *IEEE Transactions on Vehicular Technology*, 2005.

[17] E. Castillo, A.J. Conejo, R.E. Pruneda, and C. Solares, "State Estimation Observability Based on the Null Space of the Measurement Jacobian Matrix," *IEEE transactions on power systems*, 2005 (vol. 20 No 3, pp. 1656-1658).

[18] Cornell Library, online at http://www.library.cornell.edu/nr/bookcpdf/c2-6.pdf, last accessed dec. 2005.

[19] Mathworks, online at http://www.mathworks.com/moler/eigs.pdf, last accessed dec. 2005.

[20] S. K. Upadhyaya, G. S. Pettygrove, J.W. Oliveira, B. R. Jahn1, "An introduction-Global Positioning System," available at: http://calspace.ucdavis.edu/Docs/Intro-GPS-Final.pdf, last accessed jan. 2006.

[21] T. Chabot, "Integrated Navigation Architecture Analysis for Moon and Mars Exploration," S.M. thesis, MIT Dept. of Aero/Astro, May 2005.

[22] K.R. Meyer and D.S. Schmidt, "From the Restricted to the Full Three Body Problem," *Transactions of the American Mathematical Society*, 2000 (vol. 352).

[23] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Technical Report No. TR 95-041, Department of Computer Science, University of North Carolina, March 2003.

[24] P.J. Huxel and R.H.Bishop, " Navigation Algorithms for Formation Flying Missions", 2nd International Symposium on Formation Flying Missions and Technologies, Sept. 2004.

[25] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering 82* March 1960, (pp. 35–45).

[26] P.J. Huxel and R.H.Bishop, "Integrated Navigation Architecture Analysis for Moon and Mars Exploration".

[27] A. G. O. Mutambara, Decentralized Estimation and Control for Multisensor Systems, CRC Press LLC, 1998.

[28] H.F. Durrant-Whyte, M. Stevens, and E. Nettleton,"Data Fusion in Decentralised Sensing Networks," *Proc. Fusion 2001*, Montreal Canada, July 2001.

[29] T. Corazzini, "Onboard Pseudolite Augmentation for Spacecraft Formation Flying," Ph.D. Dissertation, Stanford University, Dept. of Aeronautics and Astronautics, Aug. 2000.

[30] C.-W. Park, J. How, and L. Capots, "Sensing Technologies for Formation Flying Spacecraft in LEO Using Inter-Spacecraft Communications System," *the Navigation J. of the Institute of Navigation*, Vol. 49, No 1, Spring 2002 (pp. 45–60).

[31] P. Stadter, R. Heins, et al., "Enabling Distributed Spacecraft Systems with the Crosslink Transceiver," *AIAA Space Conference and Exposition*, Aug. 2001 (Paper 2001-4670).

[32] G. Purcell, D.Kuang, S. Lichten, S.C. Wu and L. Young, "Autonomous Formation Flyer (AFF) Sensor Technology Development," *TMO Progress Report,* Aug. 1998 (42-134).

[33] J. R. Carpenter, "Decentralized control of satellite formations," *Int. J. Robust Nonlinear Control*, 2002 (12:141-161).

[34] C. Park and J. P. How, "Precise Relative Navigation using Augmented CDGPS," *proceedings of the ION-GPS Conference*, Sept. 2001.

[35] F. Busse and J. P. How, "Demonstration of Adaptive Extended Kalman Filter for Low Earth Orbit Formation Estimation Using CDGPS," *ION-GPS*, Sept. 2002.

[36] G. Bierman, Factorization Methods for Discrete Sequential Estimation, Academic Press. New York, 1977.

[37] P. Ferguson, "Distributed Estimation and Control Technologies for formation Flying Spacecraft," S.M. thesis, MIT Dept. of Aero/Astro, Jan. 2003.

[38] R. Brown, P. Hwang, Introduction to Random Signals and Applied Kalman Filtering. Third Ed., pp. 366-7, John Wiley & Sons, 1997.

[39] S. F. Schmidt, "Application of State-Space Methods to Navigation Problems," in C. T. Leondes (ed.), Advances in Control Systems, NY: Acad. Press, 1966 (Vol. 3).

[40] S. I. Roumeliotis and G. Bekey, "Distributed Multirobot Localization", in *IEEE Transactions on Robotics and Automation,* Oct. 2002 (vol. 18 No 5).

[41] http://ssl.mit.edu/spheres/, last accessed jan. 2005.

[42] S.-J. Chung, E.M. Kong, and D.W. Miller, "Dynamics and Control of Tethered Formation Flight Spacecraft Using the SPHERES Testbed," *AIAA Guidance, Navigation and Control Conference*, San Francisco, CA, Aug. 2005.

[43] J. Enright, M. Hilstad, A. Saenz-Otero, and D.W. Miller, "The SPHERES Guest Scientist Program: Collaborative Science On the ISS," *IEEE Aerospace Conference*, Big Sky, Montana, March 2004.

[44] http://ssl.mit.edu/spheres/motivation.html, last accessed jan. 2005.