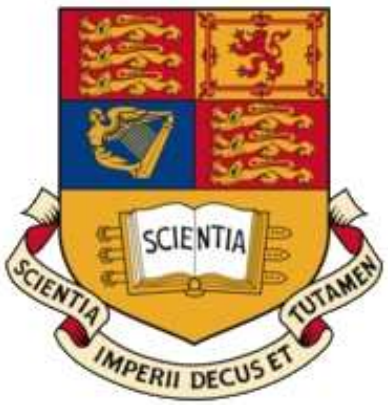
Attitude estimation of a small satellite using a multi-objective optimization approach



Department of Aeronautics
Imperial College London

E.T.S Enginyeria Industrial
Universitat Politècnica de Catalunya

AUTHOR:

Germán Gambón Bru

SUPERVISOR:

Dr. Eric Kerrigan

June 29, 2011

Abstract

The present report analyses the application of moving-horizon multi-objective optimization to the estimation of the state variables of a small satellite. The involved physical system is mathematically described by a system of seven differential equations for seven state variables, accounting for the rates of spin and the body attitude, both in body reference with respect to the fixed orbit frame.

The available input data for the estimation are noisy measures of the attitude, in the form of euler angles that are later translated to quaternions, thus avoiding singularities. The optimization problem has the differential equation system as a constraint for the variables to follow. The objective function minimized is the square norm of the error between the solution of the optimization and the attitude measurements. The problem is solved by an interior point method with the IPOPT package, being manipulated through its Matlab interface. Controlling resources is done by a moving horizon approach, which allows for only a smaller part of the measurements to be fed into the optimization. This however, displays a noticeable effect on the performance of the method, thus making an important parameter to study.

Results show the extreme importance of supplying an attitude guess close to the solution, which is easily fulfilled by the measurements. For the rates however, no guess in necessary for reasonable horizon lengths, whereas large ones do require such a guess for the solver to be able to find a good solution. Moreover, while results show difficulty in predicting the solution error for specific simulation parameters, a clear relation is found between the parameters and the computation time. Precisely, a linear relation exists between computation time and horizon length, whereas a quadratic relation is observed versus sample rates.

Advantages of this method of estimation are a high accuracy and wide working range of horizon lengths and sample rates. Disadvantages are the significant computation effort and the difficulty to bound it, both being variables that need special treatment for the present method to operate correctly in a real-time application.

Further discussion about the formerly mentioned performance indicators of the system, namely the accuracy of the solution and the computing time, will be developed throughout this document, and in which way the are related to the simulation parameters, i.e. the horizon length, the sample rate, and the initial conditions.

Acknowledgements

Here I would like to express my gratitude to Dr. Eric Kerrigan, for accepting my Master's Thesis application under his supervision.

I would also like to thank Ms Paola Falugi for all her useful advices and general willingness to help during these four months.

Many thanks to the people in Dr. Kerrigan's control group too - Mr Shakil Ahmed, Mr Stefano Longo, Mr Andrea Suardi, Mr Juan Jerez, Mr Ammar Hasan, Mr Edo Abraham, and the more infrequent attendants, for allowing me to listen to their presentations, as well as for paying attention to my own.

Finally, I like to dedicate this work to my family - Lorena, Josefina and German, for supporting me with this endeavour as they always do.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A relatively new kind of satellites of reduced dimensions are recently gaining importance in the field of outer atmosphere experiments due to their cost and development-time inexpensiveness compared to their major counterparts. The small size of these machines highly restraints the type of control mechanisms available, in favour of the necessary tools for the actual purpose of the satellite. For controlling the satellite motion, magnetic torquers have been found to be very suitable, since they agree with size, low cost, low power consumption. Using these devices however, it is compulsory that the satellite lies on low-Earth orbit. This last condition is compulsory due to its working principle, which relies on the interaction of the magnetic field of the earth with the magnetic field generated by the magnetorquers' coils. The problem with this approach is that the resulting force of the interaction is the cross product, so it can only span a 2-dimensional subspace, making it instantaneously uncontrollable. Still, the satellite can be fully controlled over time thanks to the form of the earth magnetic field and to the satellite being in orbit. We will not go into more detail on this topic since it is not the main objective of the present report or a requirement for it, but rather another module in the general control problem that can be treated separately. Further information can be found in the literature ([1]).

Regardless of the control method, accurate measures of the current attitude and rates of spin (rates onwards) are fundamental for the satellite to be controlled precisely. The former are easily obtained through common on board sensors. Rates however, are not measured, thus the essential need of an estimation method to obtain them. Therefore, the estimation problem is intended for the rates estimation plus for improving the attitude measurements.

For this task, non-linear multi-objective optimization is employed. Through this method, it is possible to input a set of non-linear differential equations representing the dynamics of the problem variables. One can also provide additional non-linear equations constraints on the final state of the variables, as well as between the

beginning and the end of the time span. Finally, there is also the objective function to be minimized with respect to the problem variables plus optional parameters. Note as well that this function has an integral part and an algebraic part. All these available settings make this kind of optimization very flexible, and therefore suitable for our purpose.

Naturally, the meaning of a good estimation is a small error with respect to the actual values, in other words, a solution as close as possible to the real trajectory. However, since we do not know those value we must assume that the measurements will be similar. This idea leads to the form of the objective function, which in this case is the square of the error norm. No other constraints are needed to find a good solution.

The optimization problem needs the measurements to find the solution. However, the length of the simulation horizon, which is the data window taken, and the distance between successive samples, or sample rate, are parameters that affect the accuracy of the solution, as well as the computing effort. A this point, we must recall the power restrictions on the satellite, which translate into fewer computing resources, thus requiring to find a balance between the optimization parameters, namely horizon length and sample rate, and the performance indicators, which are accuracy and computing effort.

Specially critical is the computation time, as the present estimation method is intended to work on-line, this is, under real-time conditions. For this reason, the duration of the calculation should always be shorter than a specific value. Along these lines, a bound on the computation time would be certainly useful, although very difficult to obtain due to the nature of the problem.

In summary, the main objective of this document is to find a relation between the optimization parameters and the impact on the performance of the optimization. For this purpose, we will develop a Matlab simulation of the estimation problem for a real trajectory of the satellite, and compare the obtained results to be able to extract conclusions on the method behaviour.

Chapter 2

Mathematical description of the physical system

The satellite in question is a CubeSat [4]. Its fundamental specifications were developed by a joint collaboration between the California Polytechnic State University and the Standford University to provide low-cost space exploration alternatives suitable for a wider audience, thus allowing training or experimentation opportunities to Universities.

The CubeSat is designed to be modular, with each module or unit size of $100x100x100$ mm, 1 kilogram of weight, and 1 litre of useful volume. Its modularity allows for quite flexible scalability so as to be able to fit the necessary devices for a particular application in a number of connected units without much increased difficulty or cost.

Current low-earth orbit experiments being carried out target scientific areas such as weather studies, atmospheric analysis, energetic particle studies, disaster monitoring, spacecraft damage studies or spacecraft attitude control methods. In fact, since these smaller brand of satellites have the same essential components as their larger siblings, i.e communication systems, processing units or solar panels, the only restrictions imposed to candidate experiments are related with the smaller size and lower power requirements. As of now, in space demonstrations of up to 3 units have been successful and 6-unit experiments are being planned.

In this section, we will summarize the mathematical model and its development. However, it will not be explained in much detail since extended discussion on this topic has been thoroughly treated in other sources.

2.1 Attitude description

2.1.1 Reference frames

In order to easily develop the motion model, it is convenient to define four different reference frames and perform the appropriate transformations when necessary. These frames are oriented as follows.

Earth-Centered Inertial frame

This is a non-accelerated reference frame in which Newton's laws are valid. The frame's center is fixed on the Earth's center, with the z-axis pointing towards the North Pole and the x-axis pointing towards Vernal Equinox. The y-axis completes the right-hand cartesian coordinate system.

Earth-centered Earth Fixed frame

It is also positioned on the Earth's center, with its z-axis pointing towards the North Pole, its x-axis towards the intersection point between the Greenwich meridian with the Equator, and its y-axis completing the right-hand system. As its name suggests, this frame is fixed on the Earth, therefore moving and rotating together.

Orbit frame

Also known as the satellite coordinate system, this frame is centered on the satellite, with the z-axis pointing towards the Earth's center, the x-axis pointing in the direction of the orbit trajectory, and the y-axis completing the right-hand system.

Body frame

This frame is fixed to the satellite. Its orientation in space is defined with respect to the Orbit frame using the *Euler angles* hence representing the satellite's attitude. Its centered is located on the satellite's center of mass and its axes point along its principal directions, perpendicularly to the satellite's surface and forming a right-hand cartesian coordinate system. Even agreeing with this orientation some freedom is still available, although the final selection will not affect the motion equations.

2.1.2 Euler angles

As mentioned before, the orientation of the Body frame with respect to the Orbit frame defines the attitude, and it is expressed in Euler angles. These are ϕ for the roll, θ for the pitch, and ψ for the yaw. At this point, we can derive the transformation matrix between these two reference frames, in order to use it on the motion equations. However, we will express this matrix in terms of quaternions rather Euler angles since that is the form we will be using on the simulations.

$$C_{b/o} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.1}$$

The angular velocity vector $\omega_{b/o}^b = [P\ Q\ R]^T$, which components are the rates, is determined by the Euler angles and their derivatives. In this case, expressing it in terms of quaternions is too complicated, so we will use a relation explicitly based on the Euler angles

$$\begin{aligned} P &= \dot{\phi} - \dot{\psi}\sin\theta \\ Q &= \dot{\psi}\cos\theta\sin\phi + \dot{\theta}\cos\phi \\ R &= \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi \end{aligned} \tag{2.2}$$

2.1.3 Quaternions

Unfortunately, Euler angles present singularities at some points in space, hence the need of an alternative representation of the attitude to avoid this problem. The Euler parameters are a quaternion representation of the Euler angles that correct the aforementioned problem. From here on, we will be referring them as quaternions $q = [q_0\ q_1\ q_2\ q_3]^T$. Following is shown the relation between these parameters

$$\begin{aligned} q_0 &= \pm(\cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ q_1 &= \pm(\sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ q_2 &= \pm(\cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ q_3 &= \pm(\cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{aligned} \tag{2.3}$$

and the inverse relations

$$\begin{aligned} \phi &= \text{atan2}(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ \theta &= \arcsin(2(q_0q_2 - q_1q_3)) \\ \psi &= \text{atan2}(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{aligned} \tag{2.4}$$

As a final remark, note that the quaternions must satisfy a unitary square sum, or

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{2.5}$$

2.2 Rotational dynamics

The rotational dynamics are derived using Newton's law for rotational motion, which expressed in Body frame coordinates, states that the sum of torques is equal to the derivative of the body angular momentum vector with respect to its center of mass. Therefore, after some transformations the resulting equation is

$$\tau_{gg}^b + \tau_c^b + \tau_d^b = J^b \dot{\omega}_{b/i}^b + \omega_{b/i}^b \times J^b \omega_{b/i}^b \tag{2.6}$$

where τ_c^b is the torque generated by the control action, τ_{gg}^b the torque caused by unalignement of the center of mass and the center of gravity, and τ_d^b the torque due to disturbances. $\omega_{b/i}^b$ is the inertial angular velocity vector expressed in Body frame, and finally J^b is the inertia tensor in Body frame, which due to the axisymmetric shape of the satellite results in a diagonal matrix of components J_x, J_y and J_z.

2.2.1 Gravity gradient torque

In Body frame, the gravity gradient torque is expressed as

$$\tau_{gg}^b = \frac{3GM_T}{|P^b|^5}(P^b \times J^b P^b) = \frac{3GM_T}{|P^b|^5} \begin{bmatrix} (J_z - J_y)P_y P_z \\ (J_x - J_z)P_z P_x \\ (J_y - J_x)P_x P_y \end{bmatrix} \tag{2.7}$$

with $P^b = [P_x \ P_y \ P_z]^T$ is the position vector to the center of mass expressed in Body frame. This can be further developed to explicitly depend on the euler angles.

$$\tau_{gg}^b = 3\omega_0^2 \begin{bmatrix} (J_z - J_y)a_{23}a_{33} \\ (J_x - J_z)a_{33}a_{13} \\ (J_y - J_x)a_{13}a_{23} \end{bmatrix} \tag{2.8}$$

where $\omega_0 = \sqrt{\frac{GM_T}{a^3}}$ is called the rate of frequency, and a being the semi-major axis of the elliptical orbit described by the satellite. Furthermore,

$$\begin{aligned} a_{13} &= -\sin\theta \\ a_{23} &= \sin\phi\cos\theta \\ a_{13} &= \cos\phi\cos\theta \end{aligned} \tag{2.9}$$

which are the elements of the direction cosine matrix.

2.2.2 Control torque

The control torque is dependent on the dipole moment, M^b, generated by the coils and the Earth Magnetic Field (EMF), B^b, in the following way

$$\tau_c^b = M^b \times B^b = \begin{bmatrix} 0 & B_z^b & -B_y^b \\ -B_z^b & 0 & B_x^b \\ B_y^b & -B_x^b & 0 \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \tag{2.10}$$

The cross product has been expanded for a simpler expression, where $B^b = [B_x^b \; B_y^b \; B_z^b]^T$ and $M^b = [m_x \; m_y \; m_z]^T$.

2.2.3 Inertial angular-velocity vector

Using this vector simplifies the rotational equations.

$$\omega_{b/i}^b = \omega_{b/o}^b + C_{b/o}\omega_{o/i}^o \tag{2.11}$$

with $\omega_{o/i}^o = [0 \; -\omega_0 \; 0]^T$ a constant value, being ω_0 the rate of frequency or satellite rotation rate, which has been mentioned before.

2.2.4 Kinematic equations

The kinematic equations representing the rigid body orientation complete the mathematical system describing the angular motion of the satellite. In this case, the system is directly expressed in quaternions, yielding the following differential equation

$$\dot{q} = \frac{1}{2} \begin{bmatrix} 0 & -P & -Q & -R \\ P & 0 & R & -Q \\ Q & -R & 0 & -P \\ R & Q & -P & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.12}$$

2.2.5 Earth Magnetic Field model

The International Geomagnetic Reference Field (IGRF) model (ref) is an accurate representation of the intensity of the Earth's Magnetic Field in space. However, it is very complex and the accuracy improvement provided with respect to a simplified version is of a 5-10%. In our satellite's motion model we will be using a first order and first degree IGRF simplified model, since it gives sufficient accuracy for the initial design.

2.3 Control model

The control model that the estimation algorithm will be working with is described by

$$\begin{aligned} \dot{x} &= f(x, u, w, t) \\ y &= h(x, u, v, t) \end{aligned} \tag{2.13}$$

representing a system of non-linear differential equations. In this description, x, u, y are the state, input, and output variables vector respectively, whereas w and v are the process and measurement noise, and t the time.

The variables of interest of the model are the three rates, and four quaternions, since together they express the attitude and motion of the system. Hence, the state variable vector will be

$$x = [P\ Q\ R\ q_0\ q_1\ q_2\ q_3]^T \tag{2.14}$$

The output variables are the rates themselves, and the inputs are three control actions, one for each coil, and are provided to the estimation algorithm by the control algorithm.

In order to find the non-linear differential equation system, the motion equations developed earlier need to be rearranged. For this, we need to expand the derivative part first

$$\dot{\omega}^b_{b/i} = \dot{\omega}^b_{b/o} + \dot{C}_{b/o}\omega^o_{o/i} = \begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} - \omega_0 \dot{v}_2 \tag{2.15}$$

where v_2 is the second column of the $C^b_{b/o}$ matrix, for simplicity. It is easy now to

solve the equations for the rates derivatives

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = (J^b)^{-1} \left[\sum \tau^b - \omega^b_{b/i} \times J^b \omega^b_{b/i} \right] + \omega_0 \dot{v}_2 \tag{2.16}$$

Regarding the quaternion derivatives, they directly come from the kinematic equations (2.12), thus completing the system of seven differential equations for the seven state variables.

Chapter 3

The estimation problem

The present problem consists in the minimization of the error between the state variables - solution of the problem - and the states measurements. It is a non-linear problem, since there are non-linear equations both in the constraints, due to the ODE system, and in the objective function, due to its form. It is also a purely discrete problem because the measurements are obtained at a certain time interval. This implies two important points, one being the discrete definition of the measurements, naturally. The second, not as obvious, stems from the sample rate choice flexibility that we wanted to imprint in the estimation algorithm. With this approach, for large sample rates with respect to the dynamics' speed, any trajectory interpolation between consecutive measurements would be very inaccurate, or simply incorrect. For these two reasons, the optimization problem will be discrete instead of a continuous time approximation.

3.1 Description of the equivalent optimization problem

The present optimization problem is a subcase of a more general form being assessed by the optimization solver, which is called IPOPT (Section 4.1). Further explanation on it full potential can be found in [3].

$$\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x},\mathbf{u}) \quad \triangleq \quad \min_{\mathbf{x},\mathbf{u}} \quad \sum_{i=0}^{N-1} L_i(\tilde{x}_i, u_i) \tag{3.1}$$

subject to

$$\begin{aligned} \tilde{x}_{i+1} &= \phi(\tilde{x}_i, u_i) \\ g(\mathbf{x},\mathbf{u}) &= 0 \end{aligned} \tag{3.2}$$

The estimated states vector (just states vector onwards) is $\mathbf{x} \triangleq [\tilde{x}_0, \ldots, \tilde{x}_{N-1}, \tilde{x}_N]^T$ and the control inputs $\mathbf{u} \triangleq [u_0, \ldots, u_N]^T$. The states follow the trajectory defined by the differential equation system $\dot{\tilde{x}} = f(t; \tilde{x}, u)$, however since the measurements y to which the states are compared (3.3) are not continuous, it is necessary to find a discrete version of the ODE. This can be interpreted as a map function, $\phi(\tilde{x}_i, u_i)$, obtained from solving the ODE system $\dot{\tilde{x}} = f(\Delta; \tilde{x}, u)$ with initial condition $\tilde{x}(0) = \tilde{x}_i$ and timespan Δ, which is actually the sample rate of the measurements. The reason behind this approach is that integrating the system step-wise effectively yields its implicit discrete version, which is very convenient in this case since the problem size is quite significant and supplying the analytical form of it would be difficult. Another advantage given by this solution is that first order derivative information of the differential system can be found without much more computing effort using the concept of sensitivities, explained further below.

The function $g(\cdot)$ is the equality constraints function and contains the continuity constraints for the discrete ODE system.

For this problem, a moving horizon scheme is used to control the amount of data fed into the optimizer. This scheme is characterized by three parameters that relate to each other. One is the horizon length H, or time interval from the current instant in time to the last one to be used. Another, the sample rate Δ. And the last, the number of samples N taken. At this point, we can define the relation among them, $N \triangleq \frac{H}{\Delta}$.

3.2 Cost function

The cost function $J(\mathbf{x}, \mathbf{u}) \in \mathbb{R}^1$ consists in the stage cost term $L(\mathbf{x}, \mathbf{u})$, sum of $L_i(\tilde{x}_i, u_i)$. For our case, a suitable form of this function is a scalar indicator of the error, hence we use the square error norm. Mathematically,

$$L_i(\tilde{x}_i, u_i) \triangleq \|p(\tilde{x}_i) - y_i\|_2^2 \tag{3.3}$$

subject to

$$y_k = q(x(k\delta)) \qquad k = 0, 1, \ldots, N-1$$

where $p(\tilde{x})$ is a function that extracts the quaternions from the estimated states $\tilde{x}$, and $y \triangleq [y_0, \ldots, y_{N-1}]^T$ is the measurements vector obtained from the unknown actual states x, with added noise represented by the function $q(x)$.

3.3 Providing the differential equations system

Next, for convenience purposes let us define $z \triangleq [\tilde{x}_0, u_0, \tilde{x}_1, u_1, \ldots, \tilde{x}_{N-1}, u_{N-1}, \tilde{x}_N]^T$. The function constraints of the model are included in the constraints vector $c(z)$

$$c(z) = \begin{bmatrix} \tilde{x}_0 - \tilde{x}(t_0) \\ g(z) \end{bmatrix} \tag{3.4}$$

where $g(z)$ is the equality constraints function, which contains the continuity restrictions needed for the state variables to be appropriately merged between the consecutive boundary value problems generated by the way in which the map function is calculated.

$$g(z) \triangleq \begin{bmatrix} \tilde{x}_1 - \phi_0 \\ \vdots \\ \tilde{x}_N - \phi_{N-1} \end{bmatrix} \tag{3.5}$$

Consequently, $C(z) \in \mathbb{R}^{7(N+1)}$.

In order to obtain a flexible description of the equations that can be easily modified to use a different direct collocation method, the above function is arranged so that

$$g(z) = AV_x z + B\Phi(z) \tag{3.6}$$

with V_x a matrix that extracts the states $\tilde{x}_i$ with $i \neq 0$, from vector z, and $\Phi(z) \triangleq [\phi_0, \ldots, \phi_{N-1}]^T$. Note that this convenience definition is specially useful for the few changes made to ICLOCS [6] (Section 4.1) to comply with the current structure of the code.

3.4 First order derivative information

The solution to the optimization problem relies on a summarizing function $\mathcal{L}(z, \lambda)$ called the Lagrangian.

$$\begin{aligned} \mathcal{L}(z, \lambda) &\triangleq J(z) - \lambda^T C(z) \\ &= L(x, u) - \lambda_0^T (x_0 - x(t_0)) - \sum_{i=0}^{N-1} \lambda_{i+1}^T (x_{i+1} - \phi_i) \end{aligned} \tag{3.7}$$

The components of vector $\lambda \triangleq [\lambda_0, \ldots, \lambda_{N-1}]^T$ are parameters with no physical meaning, only needed by the optimization algorithm to find the solution. For the cost function to be minimized, the first derivative of $\mathcal{L}(z, \lambda)$ with respect to both λ and z must be zero. Moreover, to be able to differentiate minimums from maximums and inflection points, the second derivatives have to be calculated as well. Hereafter will be explained the different parts in more detail.

3.4.1 Objective function derivative

The gradient of the cost

$$\begin{aligned} \nabla J(z) &= \begin{bmatrix} \frac{\partial L(z)}{\partial z_0} & \cdots & \frac{\partial L(z)}{\partial z_{N-1}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial L(z)}{\partial \tilde{x}_0} & 0 & \cdots & \frac{\partial L(z)}{\partial \tilde{x}_{N-1}} & 0 \end{bmatrix} \end{aligned} \tag{3.8}$$

can be expressed explicitly owing to the simplicity of its primitive, yet it can also be evaluated through finite differences with minor added computing effort. Also note that $\nabla J(z) \in \mathbb{R}^{7 \times N}$

3.4.2 Constraints vector derivative and sensitivities

The Jacobian of the constraints is a matrix of $\mathbb{R}^{7(N+1) \times 7N+10}$

$$\begin{aligned} \nabla c(z) &= \begin{bmatrix} I \\ \nabla g(z) \end{bmatrix} \\ &= \begin{bmatrix} I & & & & \\ -\frac{\partial \phi_0}{\partial \tilde{x}_0} & 0 & I & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\partial \phi_{N-1}}{\partial \tilde{x}_{N-1}} & 0 & I \end{bmatrix} \end{aligned} \tag{3.9}$$

and depends on the sensitivities s_i which need to be calculated numerically for coming from an implicit function. For this task two approaches can be taken. On the one hand, it is possible to evaluate the sensitivities using a finite difference approximation

$$s_i \triangleq \frac{\partial \phi_i}{\partial \tilde{x}_i} \approx \frac{\phi(\tilde{x}_i + e, u_i + e) - \phi(\tilde{x}_i - e, u_i - e)}{2e} \tag{3.10}$$

this however, implies solving the ODE system twice for every s_i in order to get the value of the implicit function at the given points. On the other hand, the sensitivities can be obtained along with the ODE system using a solver with such a built-in function. Sundials' CVODES package [10] (Section 4.1) has been selected for this purpose. Moreover, this alternative is significantly faster since both calculating the implicit function and the sensitivities can be done not only simultaneously but also just once for each Newton iteration of the optimization algorithm (Section 4.1).

In fact, the aforementioned sensitivity concept employed by CVODES is more general than the definition that has been given. The solver lets the user define the so called sensitivities in the way that best suits the application, as long as it agrees with the following assumptions. For a differential equation system of the form

$$\begin{aligned} \dot{y} &= f(t, y, p) \\ y(t_0) &= y_0(p) \end{aligned} \tag{3.11}$$

where $y = \phi(t; p)$, or in other words, ϕ is the function obtained by solving the differential system, with p a vector of parameters. The sensitivities are defined as $s \triangleq \frac{\partial y}{\partial p}$, and these can only be calculated indirectly by solving the equation obtained through its time derivative

$$\dot{s} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial y(t)}{\partial p} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial}{\partial p}\phi(t;p) = \frac{\partial}{\partial p}f(t, \phi(t;p), p) = \frac{\partial f}{\partial \phi}\frac{\partial \phi}{\partial p} + \frac{\partial f}{\partial p} \iff$$

$$\dot{s} = \frac{\partial f}{\partial y}s + \frac{\partial f}{\partial p} \qquad s(t_0) = \frac{\partial y_0}{\partial p} \tag{3.12}$$

In the context of our problem, the above equations can be easily translated and still agree with the development. However, it should be noticed that the mapping function ϕ is discrete and therefore not continuous-differentiable, yet close to it depending on the sampling rate and the speed of the system's dynamics. In fact, studying the relation between the ability to find an accurate solution of the optimization and the sampling rate is one of the objectives of the present thesis. This relation is shown experimentally instead of analytically however, since should the latter be even possible, it is out of the scope of this study due to the currently unreasonable requirements.

Then, in our case, the differences are

$$\begin{aligned} y(t) &\triangleq x(t) \\ y_i(0) &\triangleq \tilde{x}(t_i) = \tilde{x}_i \\ p_i &\triangleq z_i = [\tilde{x}_i \; u_i]^T \\ s_i &= \frac{\partial \phi(\Delta; \tilde{x}_i, u_i)}{\partial z_i} \end{aligned} \tag{3.13}$$

$$\dot{s}_i = \frac{\partial f}{\partial x} s_i + \begin{bmatrix} 0 & \frac{\partial f}{\partial u_i} \end{bmatrix} \qquad s_i(t_0) = \frac{\partial \tilde{x}_i}{\partial z_i} = [I_n \; 0] \tag{3.14}$$

where subindex i represents the current step and x_i, u_i are the initial conditions of the differential system at i. Additionally, it is assumed that $y_{i+1} = \phi(\Delta; \tilde{x}_i, u_i)$ which only holds true when $\Delta \to 0$, still it can be an accurate approximation for adequate values of Δ. Further explanation on this topic can be found on the CVODES user guide [8], and on the multiple-shooting section in [11].

As equation 3.14 suggests, the derivatives of the ODE system with respect to the state variables need to be calculated. These, again, can be obtained either numerically through finite differences, or analytically, preferably using a symbolic mathematics software due to the size of the system. Let us remark that both methods have been tried on the simulations, showing no significant difference either in computing effort or accuracy.

3.5 Second order derivative information

The Hessian of the Lagrangian $\nabla^2 \mathcal{L}$ calculation is the most time consuming task in each optimization iteration. Currently, there are two approaches to obtain it. One of them requires the approximation of $\nabla_z^2 \phi$ by finite differences, and therefore solving the ODE system plenty of times, not unlike in (3.10). The other method relies on an approximation of the Hessian by a limited-memory quasi-newton approximation using the L-BFGS method, which is already implemented in IPOPT. This procedure is most appropriate in cases where calculating the derivatives either numerically or analytically is not feasible. Since that is the case in the present situation, the latter alternative is the one being used. More comprehensive insight into the general behaviour of the optimization algorithm, as well as the available transcription methods and related comments on this topic, can be found in [2], [11] and [5].

Chapter 4

Simulations implementation

Hereafter are explained in detail the tools, considerations and operative procedures that have been used during the implementation of the estimation problem and the optimization problem that the former is based on.

4.1 Matlab packages used for the simulation

Three software packages have been employed within Matlab, accounting for the optimizer, the control-problem setup interface, and the ODE solver.

IPOPT

The solver used for the optimization problem is based on an interior point method, hence its name, Interior Point Optimizer (IPOPT). This software package is capable of solving non-linear optimization problems of the present type, and even others of a more general form. Its full capabilities can be found on [3]. However, the solutions found by this software correspond to local minima, therefore good initial guesses are important to improve the algorithm robustness. It has many bindings to different languages, including a Matlab interface, which we have been using.

ICLOCS

The Imperial College London Optimial Control Software (ICLOCS) [6] is a package of Matlab functions and scripts that provide an easy interface to set and solve optimal control problems using IPOPT. It is specially useful as this type of problems share similar characteristics that can be encapsulated and generalized in order to significantly speed up the setup process.

The current version implements a number of direct collocation transcription methods, plus a multiple shooting one with continuous time cost functions. Unfortunately, the present case uses a variation of the multiple-shooting transcription scheme with a discrete time cost function, so it has been necessary to implement a few changes to comply with the previous section development.

CVODES

This is a software for solving stiff and non-stiff ODE systems given in explicit form with sensitivity analysis capabilities [10]. It is developed by Sundials and has an interface to use it within Matlab. It will be used to obtain the discrete ODE system along with its sensitivities, by providing the software with the continuous time ODE system, plus its first derivative with respect to the state variables, and the sensitivities' initial conditions.

4.2 Obtaining the measurements

Testing the correctness of the method together with the performance analysis is only be possible if there is a way to retrieve the actual trajectory of the satellite. Under laboratory conditions, the easiest solution to this consists in eliminating the control action altogether, and simply integrating the ODE system using a numerical method. Despite including the input action on the above explanations, we have not used any control input in the simulations, for the aforementioned reason. In fact, thanks to its inclusion in the theoretical development, this choice will not cripple the validity of the results when the control actions are present.

When comparing simulations with different parameters, it is fundamental to ensure that as many variables as possible remain fixed [7]. Regarding measurements, this translates into all simulations sharing the same measurement data, with the same noise characteristics. To clarify this matter, let us explain the process followed to obtain the reference.

1. Retrieve the ODE solution at equispaced instants in time
2. Add white noise to the Euler angles' trajectories
3. Provide a suitable subset of the data depending on the simulation parameters

4.2.1 Solving the ODE

The ODE is integrated normally, using common numerical methods with a reasonable small error. Next, only points of the solution at constant intervals are chosen. This interval defines the smallest sample rate, and thus it should be carefully selected. In this step we store the trajectory later used to make comparisons with the solutions.

4.2.2 Adding noise

Since the differential system is expressed in quaternions, but the measurements are based on the Euler angles in the satellite, converting from the former to the latter is necessary to follow reality as closely as possible. This transformation is quite straightforward, using the relations seen on 2.3 and 2.4, plus an added unwrapping step to extend the trigonometric functions output domain to all space, as seen in the example figure. Without this step, the angle values would be forced to fit in the function image domain, therefore spoiling subsequent quaternion reconversion.

4.2.3 Retrieving the appropriate measurements

Depending on the sample rate of the current simulation, a different subset of data from the measurements will be selected. For this to be valid however, the sample rate of the simulation must be a multiple of the measurements sample rate, chosen in the first step.

Following this procedure ensures that all simulations using the same ODE system initial conditions will share the actual trajectory and the noise source. Another advantage is that the first two steps, for creating the trajectory and the measurements, need to be performed only once.

4.3 Moving horizon implementation

The horizon, in seconds, can be understood as a window filter applied on the measurements vector, that only selects the data inside its bounds. Thus, setting up successive simulations is a matter of shifting the horizon one step forward. On the other hand, since the horizon spans from the current instant in time to a number of seconds before, only the last values of the solution vector need to be stored, as the

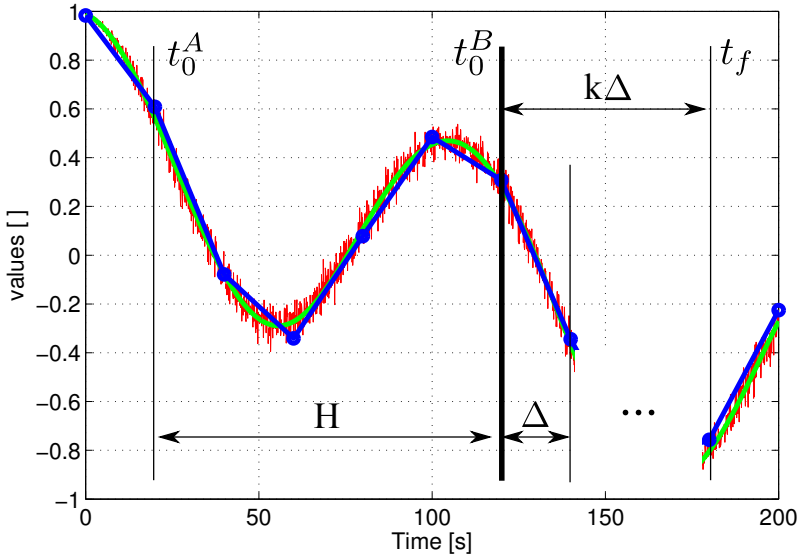


Figure 4.1: Moving horizon implementation

others have supposedly been stored in previous steps. For the first horizon of the simulations however, the whole solutions vector has been saved in order to have a better understanding of the optimizer behaviour. Note also that, for simplicity, all simulations begin with enough past information so as to fill the input data vector of the first horizon completely, thus eliminating any boundary problems.

Considering the explanation above, two slightly different simulation approaches have been chosen. Option A, which presents a horizon tail that is fixed in time, t_0^A in Figure 4.1. In this way, the first input data point remains equal for all simulations of this type. Option B however, is characterized by fixing the initial time of the simulation, t_0^B in Figure 4.1, where the first input data point changes depending on the current horizon length.

Along with the fixed parameter, the figure shows the relation between initial times $t_0^{(\cdot)}$, final time t_f, horizon length H, sample rate Δ, and number of steps k, which for our simulations is equal to 10.

4.4 Rates initial guess estimation

The rates expressions in Eq. 2.2 consist in a combination of the Euler angles and its derivatives over time. Then, since discrete measurements of the Euler angles are known, it is possible to approximate its derivatives. Due to the measurement

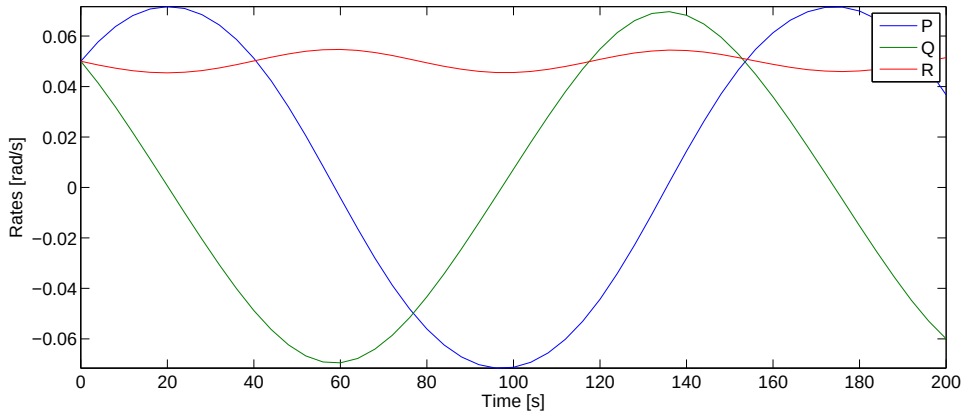
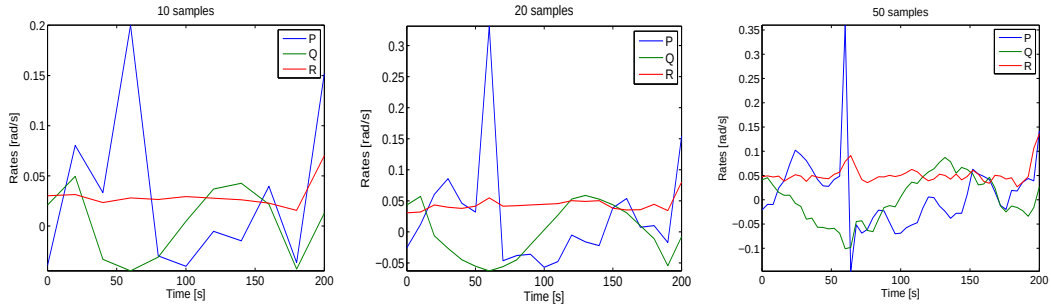


Figure 4.2: Rates actual trajectory



(a) 10 samples (b) 20 samples (c) 50 samples

Figure 4.3: Rates initial guess

noise, however, it is essential to use a noise-robust differentiator [9] so as to obtain useful information.

After some experimentation, we have chosen a differentiator of second tangency order, providing exact derivative information until second degree, plus a length of 5 samples and no more considering that the smallest amount of samples in a simulation is 10. Its performance is shown in Figure 4.3 for the three possible samples number.

Apparently, the current method shows weak performance at the end of the guess interval possibly due to boundary problems caused by the differentiator. Additionally, there is a significant singularity probably being the main reason of the computation time increase that is shown later.

Some improvements could be added to the present method, by adapting the length of the differentiator according to the length of the input, or by adding an appropriate smoothing filter. Nevertheless, we have achieved sufficiently good per-

formance with the original algorithm, therefore ignoring any further improvements.

Chapter 5

Results

Throughout this section we will be showing and comparing results between simulations, with varying horizon lengths, sample rates, initial times, and rates guesses. Note that the same ODE solution is used for all simulations.

5.1 Description of the settings and parameters

5.1.1 Satellite parameters

The parameters of the satellite in question to be used in the attitude estimation problem are summarized in Table 5.1.

Parameter	Value	Units
X-axis length	$L_x = 0.1$	m
Y-axis length	$L_y = 0.1$	m
Z-axis length	$L_z = 0.3$	m
X-axis inertia	$J_x = 0.023001$	$\text{Kg} \cdot \text{m}^2$
Y-axis inertia	$J_y = 0.023535$	$\text{Kg} \cdot \text{m}^2$
Z-axis inertia	$J_z = 0.0041965$	$\text{Kg} \cdot \text{m}^2$
Rotation rate	$\omega_0 = 0.0010730$	$\text{Kg} \cdot \text{m}^2$

Table 5.1: Satellite parameters

5.1.2 Initial conditions

The initial conditions for the integration of the ODE are

$$x(0) = [0.05\ 0.05\ 0.05\ 0.9893\ 0.0789\ 0.094\ 0.0789]^T \tag{5.1}$$

with the rates in rad/s and the quaternions equivalent to $\phi = 10^o$, $\theta = 10^o$, $\psi = 10^o$, in degrees. These setting yield the trajectory shown in Figure 5.1 for the Euler angles, and Figure 4.2 for the rates. Notice that only a suitable time range is shown for increased clarity.

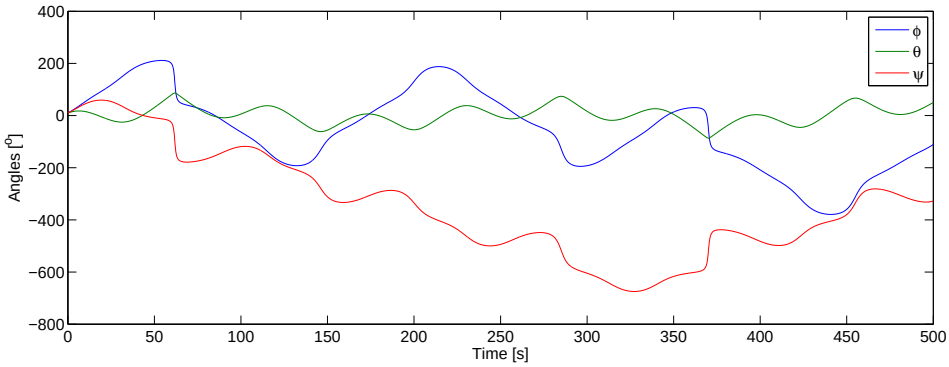


Figure 5.1: Euler angles actual trajectory

The initial conditions for the simulations however, can be any point in the trajectory given by the above ODE initial conditions, thus allowing for greater flexibility. Therefore, they are uniquely determined by the instant where they occur in the trajectory.

Regarding the added noise to the Euler angles, white noise with a standard deviation of 5^o has been employed. Converting these to quaternions a posteriori yields the quaternion trajectory shown in Figure 5.2.

5.1.3 Structure of the simulations

Independently of the simulation type or initial time, a number of standard simulations have been arranged in the following manner (Table 5.2).

With these settings, four different simulations have been run, with each of them having two variations either using or ignoring the rates initial guess. This totals a number of 8 standard simulations actually. Moreover, two additional non-standard simulations have also been analysed, presenting horizons of 200 s and 500 s, both

H[s]	N[−]	Δ[s]
10	10	1
	20	0.5
	50	0.2
20	10	2
	20	1
	50	0.4
30	10	3
	20	1.5
	50	0.6
40	10	4
	20	2
	50	0.8

H[s]	N[−]	Δ[s]
60	10	6
	20	3
	50	1.2
80	10	8
	20	4
	50	1.6
100	10	10
	20	5
	50	2

Table 5.2: Standard simulations structure

Identifier	Type	Pattern	Initial time	Variation
St0	Type A	Standard	$t_0^A = 0$ s	St0g - rates guess
St100	Type B	Standard	$t_0^B = 100$ s	St100g - rates guess
St155	Type B	Standard	$t_0^B = 155$ s	St155g - rates guess
St210	Type B	Standard	$t_0^B = 210$ s	St210g - rates guess
Sh200	Type B	$H = 200$ s $N = 10,\ 20,\ 50$	$t_0^B = 500$ s	Sh200g - rates guess
Sh500	Type B	$H = 500$ s $N = 10,\ 20,\ 50$	$t_0^B = 500$ s	Sh500g - rates guess

Table 5.3: Simulations description

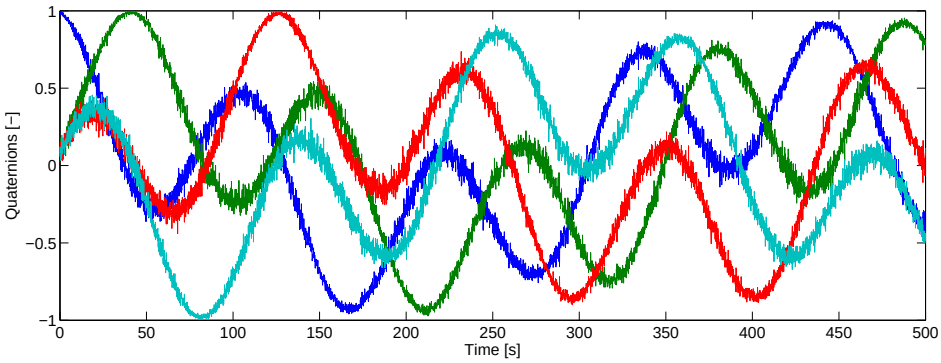


Figure 5.2: Measurements expressed in quaternions

using the pattern followed by N shown in Table 5.2. A summary of these is found on Table 5.3.

In this way, we are able to cover a wide range of parameters and yet allow for comparisons of performance as valid as possible.

5.2 Performance analysis

The present section will discuss the results obtained by means of a series of variables intended to summarize the most important aspects of the estimation behaviour.

5.2.1 Variables of interest

Worst case error norm

The most important accuracy index is the maximum error committed by the estimation solution, as it hints information a bound on the aforementioned error. Also note that a scalar value is most useful, hence the use of the norm in the error vector, which originally has seven components at each instant in time.

$$\varepsilon_{max} = \sup \|x_i - \tilde{x}_i\|_2 \qquad i = 0, \ \ldots, \ N-1 \tag{5.2}$$

Worst case computation time

The computing effort is measured through this variable, T_{max}. This is the most important performance index, along with the worst case error norm. Unfortunately, due to the fact that the simulations have been run in a common purpose operative system, user-independent multitasking during simulations can significantly change the computation time between two different runs with the exact same parameters.

Despite trying to avoid any multitasking during simulations, the previous explanation still holds true, hence forcing to only extract relative conclusions from the present performance index.

Average error norm

Not as important as the worst case, yet still interesting to observe the error deviation, and thus the estimation method robustness.

$$\hat{\varepsilon} = \sum_{i=0}^{N-1} \frac{\|x_i - \tilde{x}_i\|_2}{N} \tag{5.3}$$

Average computation time

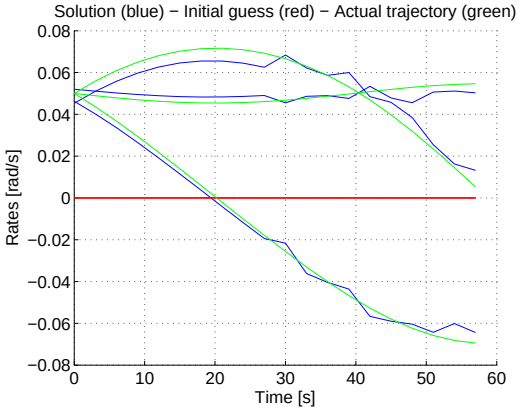
Similar in purpose to the variable right above, but regarding the computation time, $\hat{T}$.

5.2.2 Solution plots

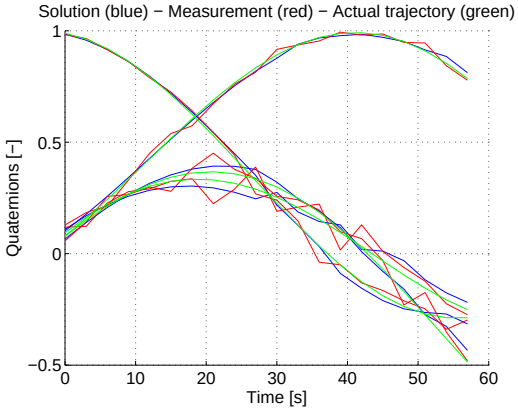
Only a few plots of the solution will be shown here, since this is enough to understand the behaviour of the method.

The plots in Figure (5.3) present three different colors encoding the following information. Blue represents the solution from the estimation problem, green the actual trajectory that the solution should track, and red for the initial conditions given to the solver.

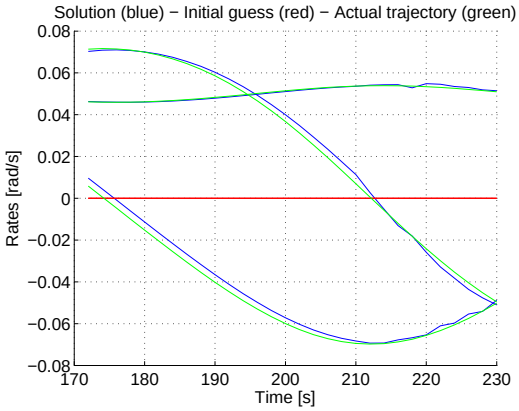
The examples chosen show the wide range of parameters used in the simulations. From top to bottom, horizon and sample rate increase, showing rates to the left and quaternions to the right. Moreover, the type of simulations change as well, as the initial time does. However, no initial guess for the rates is given in any of these cases, which judging from the goodness of the results, apparently does not affect the robustness of the method at their respective horizon-length scale.



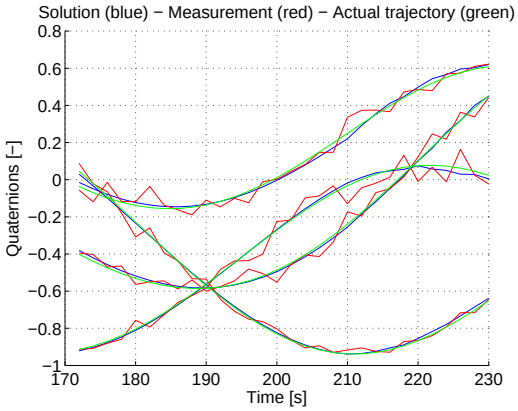
(a) St0 rates. $H = 30\,\mathrm{s}$, $N = 10$



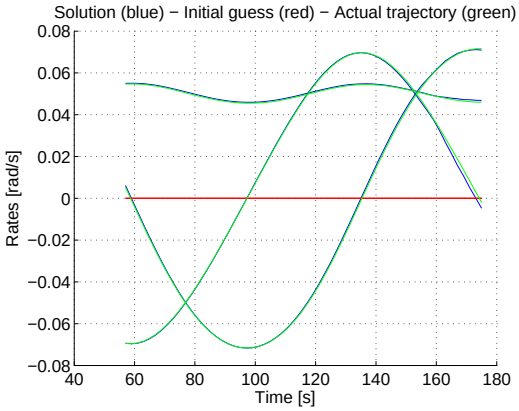
(b) St0 quaternions. $H = 30\,\mathrm{s}$, $N = 10$



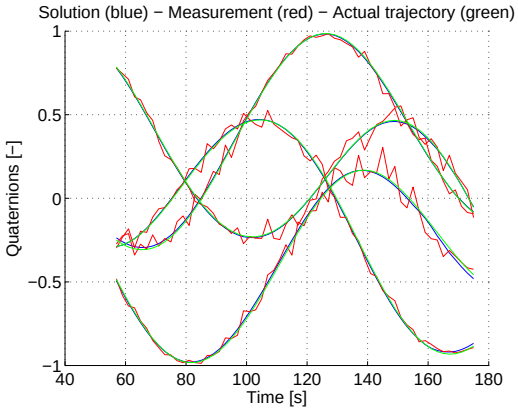
(c) St210 rates. $H = 40\,\mathrm{s}$, $N = 20$



(d) St210 quaternions. $H = 40\,\mathrm{s}$, $N = 20$



(e) St155 rates. $H = 100\,\mathrm{s}$, $N = 50$



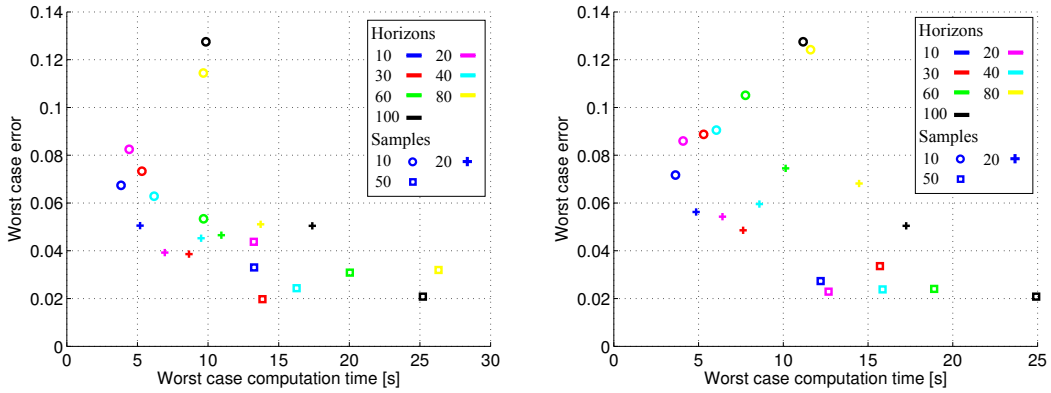
(f) St155 quaternions. $H = 100\,\mathrm{s}$, $N = 50$

Figure 5.3: Examples of solution plots

By further inspecting Figure (5.3), one notes an apparent increase in accuracy in the vertical direction of the plot array. Whether this behaviour is due to the horizon increment or the sample rate decrement, however, is studied next.

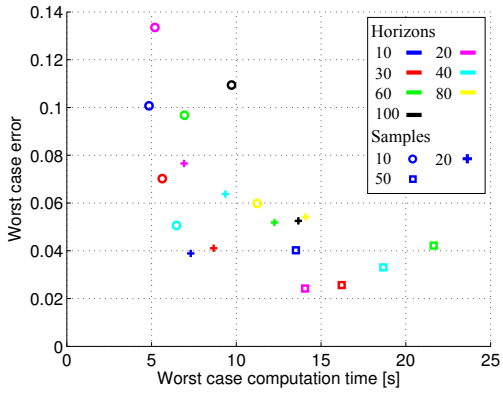
5.2.3 Pareto plots

Two-dimensional Pareto plots offer a clear way of visually representing the relation among a data set and two different variables. Its usefulness can be further improved by grouping regions of data employing different shapes and colors, thus effectively increasing the number of displayed variables by two.

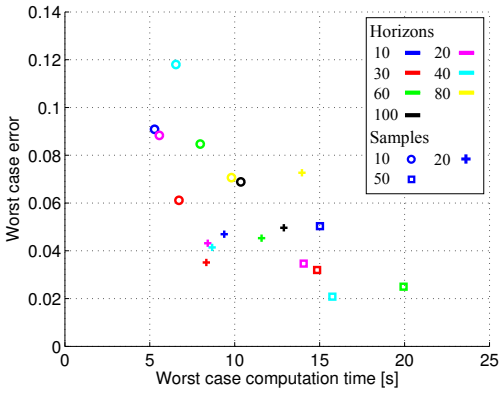


(a) St0

(b) St100



(c) St155



(d) St210

Figure 5.4: Worst cases Pareto plots

In the present case, we have included two sets of Pareto plots representing the

worst case performance variables described in Section 5.2.1, and the their average. For clarity, each Pareto plot shows only one simulation, and for cohesion reasons, worst case and average versions are not mixed together. Additionally, in order to include the horizon length H and the sample rate Δ, the data is grouped by color and shape respectively. Let us also remark, that no rates initial guess is given.

Figure 5.4 shows the worst cases Pareto plots. The first tendency to note is the impact of the number of samples on the performance. As N increases, so does the computation time, whereas the error decreases. In other words, feeding more points into the solver improves accuracy, but increases the computation time as well. Admittedly however, this is the expected behaviour, although it is not possible to extract a reliable tendency.

Regarding the horizon length, finding a tendency is more difficult due to the general variability in the position of the points with the same color. Nevertheless, simulations with large horizons appear to be more costly in terms of computation effort, whereas shorter horizons with same sample rate display similar accuracy at lower computation times. In general, the performance of large horizon simulations seems weaker than those with intermediate horizons.

A possible reason for a lower performance with large horizons might be the bigger amount of minima in the optimization problem, basically due to a longer time interval. On the other hand however, a very short horizon length does not always appear to improve the accuracy, although the computing time is clearly reduced. In this case, the reason could be insufficient information on the system dynamics for the solver to find a more accurate solution.

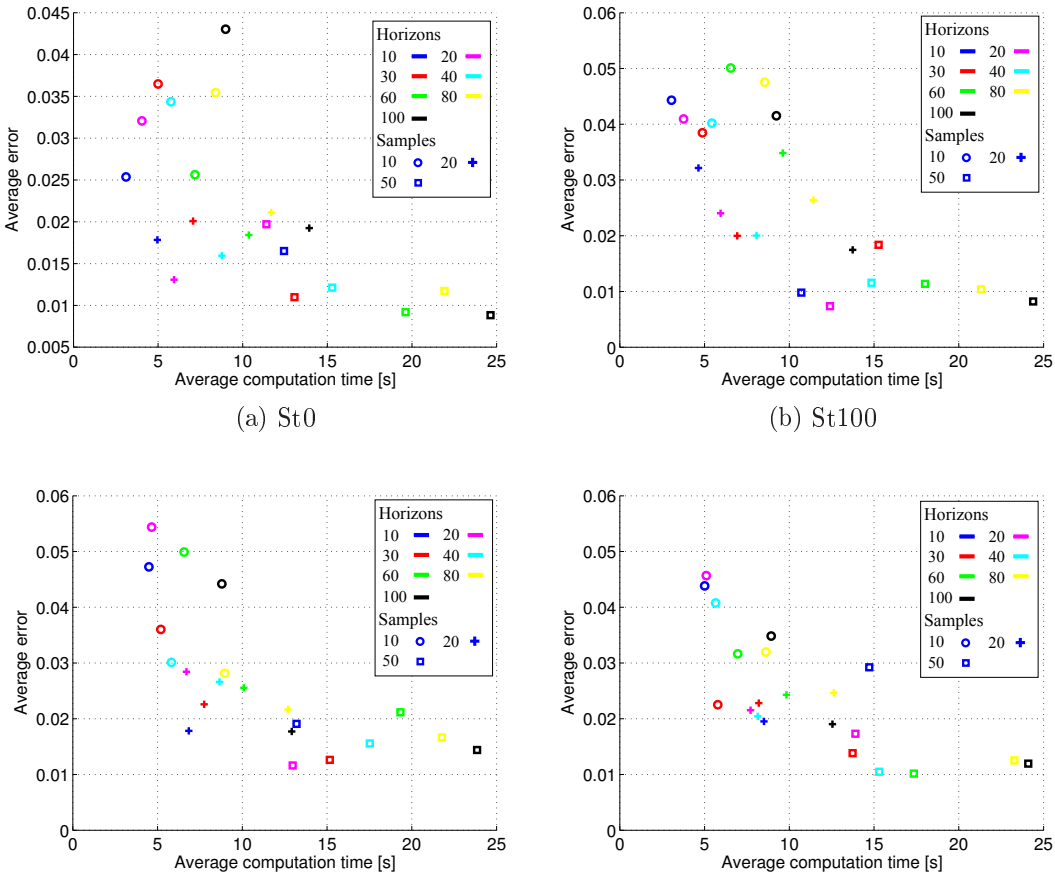
Finally, from an overall performance point of view, considering that the simulations with best performance are the ones with lower distance to the bottom left corner of the plots, intermediate values of the parameters are the best combination, this is, $H = 10 - 40s$ and $N = 20$.

Similarly, Figure 5.5 shows the average performance variables. What has been explained before applies for these as well. Even though the distribution of the points is slightly different from the worst cases plots, it is not as significant as to question the validity of the observed characteristics. Apart from this, there is not much more to mention.

5.2.4 Sorted sequence plots

With this type of plot, we are able to show almost the same information as in the Pareto plots, but from a different perspective that is more clear for observing other tendencies unnoticed yet.

These plots display one performance variable against the different horizons and sample rates. More specifically, horizons are sorted in ascending order from left



(a) St0 (b) St100

(c) St155 (d) St210

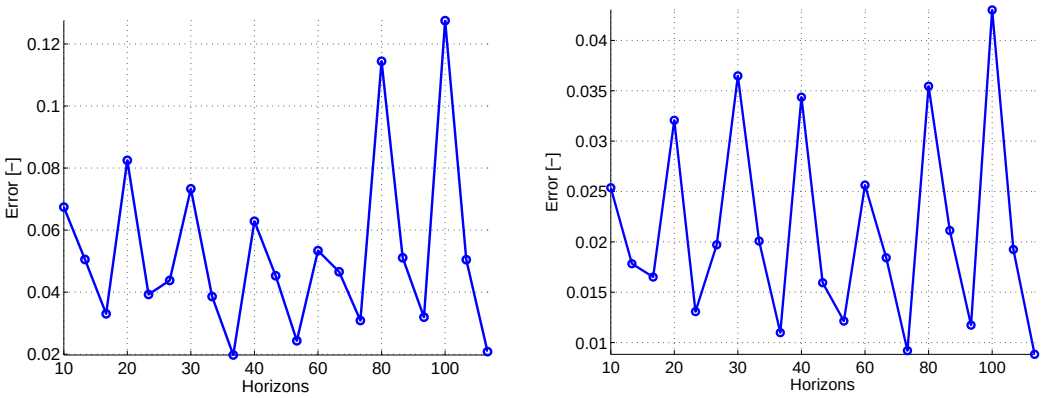
Figure 5.5: Averages Pareto plots

to right, whereas N (number of samples) is sorted in descending order inside each horizon group. Therefore, points are sorted in groups of three descending sample rates corresponding to their respective horizon. In other words, the pattern they follow is given by the standard simulation pattern, in Table 5.2.

In Figure 5.6, sorted sequence plots of the simulation St0 are shown. From this plots, a few interesting conclusions can be extracted.

Agreeing with what has been mentioned in the previous section, the error decreases with the sample rate. Along these lines, the rate at which this happens appears to decrease, as seen from the decreasing tendency until $H = 60$. Unfortunately, although the reason for this is probably related with the rates dynamics, we have not been able to discover it.

A very important characteristic is the variability of the analysed performance



(a) Worst case errors

(b) Average errors

Figure 5.6: Sorted sequence plots of simulation St0

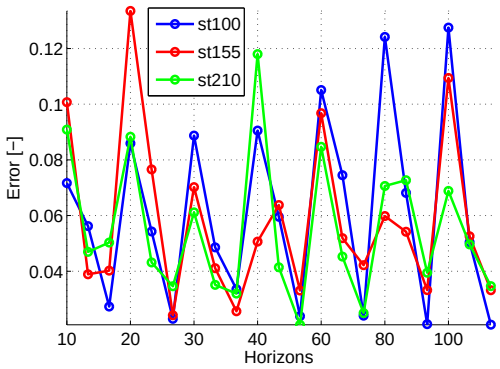
index, since it is a good measure of the robustness. As observed from the figure, both the worst case and the average plots display points with $N = 20$ quite close to each other vertically. Hence, their variability is low. In this case, a strong conclusion can be extracted, which is the significant independence of the accuracy with respect to the horizon length.

Figure 5.7 shows stacked sorted , sequence plots of simulations St100, St155, and St210. Regarding the error plots, the lowest variability in this case appears to be for $N = 50$, although points with $N = 20$ exhibit good accuracy too, as well as fairly low variability again. Unlike before, the observed spikes are more unpredictable, hence little information can be extracted in this regard.

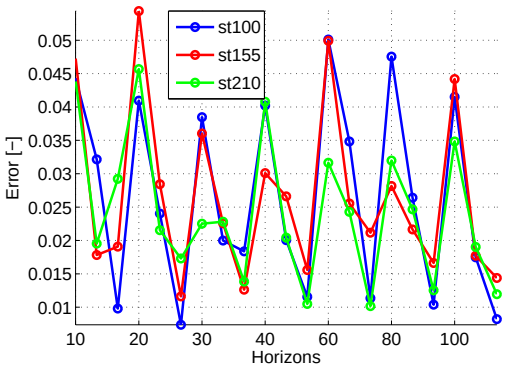
The computation time plots show a strong tendency, as opposed to the previous ones. Again, there is a clear tendency in increasing computation effort as the sample rates decreases, which can be noticed by observing that the computation time increases in groups of three, which parameters follow the standard pattern structure, in Table 5.2. Additionally however, there is another clear tendency showing increasing computation effort as the horizon length increases, i.e. moving to the right.

For further analysing these tendencies, horizon length and sample rate have been decoupled resulting in the plots shown in Figure **??**. More specifically, in Figure 5.8a the horizon length has been fixed to 60 seconds, whereas the sample rate varies freely. From this plot, it is quite clear that a relation of 2nd or higher degree exists between worst computation time and sample rate. On the other hand, in Figure 5.8b the horizon is the varying parameter and the sample rate is fixed to 3 seconds. In this case, the relation appears to be linear.

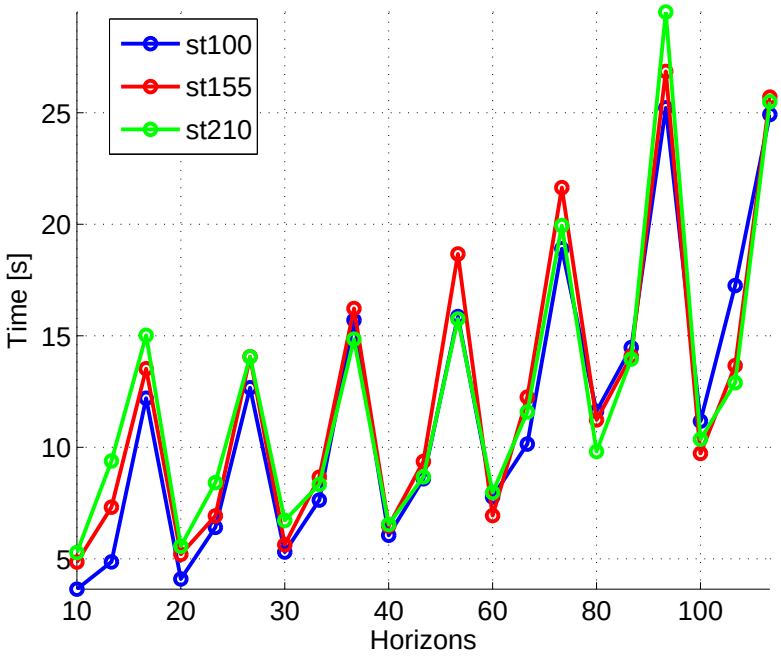
Even though these development is only based on a St155 simulation, and only



(a) Worst case errors



(b) Average errors

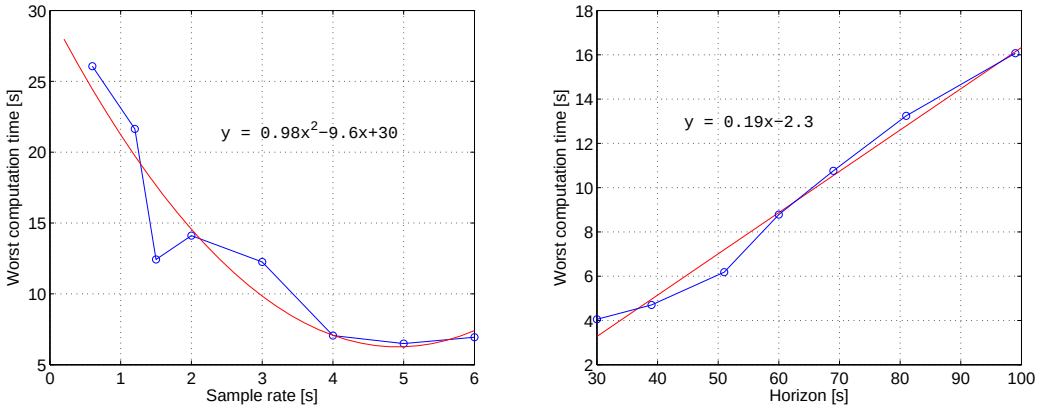


(c) Worst case computation times

Figure 5.7: Sorted sequence plots of type B simulations

one value of each parameter is investigated, the extracted conclusions are further supported by the plot shape continuity observed in Figure 5.7c.

So far, no simulation with rates initial guess has been included mainly because the accuracy obtained is exactly the same, independently of the guess. On the other hand however, computation effort is not independent, as shown in Figure



(a) WC comp. time - Δ. For H=60 s

(b) WC comp. time - H. For Δ=3 s

Figure 5.8: Decoupled horizon and sample rate effect on the computation time

5.9.

Except from simulation St0, it can be seen that, for large horizon lengths, providing an initial guess for the rates generally helps the solver find the solution faster, specially for the smaller sample rates.

5.2.5 Solutions for very large horizon lengths

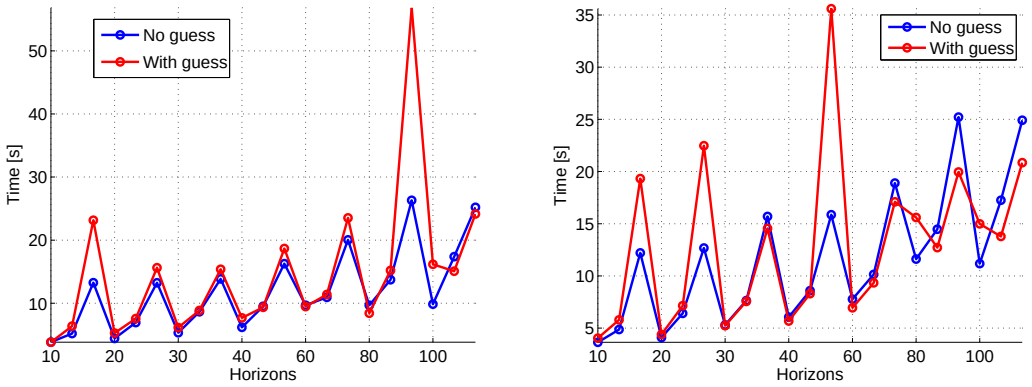
Two simulations with very large horizons have been run as well, with identifiers Sh200 and Sh500, together with their rates initial guess variation.

Results show that it is absolutely necessary to provide the initial guesses, as otherwise, the solver is not able to find a good solution. Therefore, we will only show results involving the simulation variations Sh200g and Sh500g.

The performance indicators of these simulations are shown in Table 5.4. First, regarding the errors, one can notice that, even though Sh200g displays an anomaly in ε_{max} for $N = 20$, the solutions found are correct and follow the conclusions explained earlier. On the other hand, Sh500g shows extremely large errors and computation times, since its solutions are completely wrong, except for the $N = 50$ case.

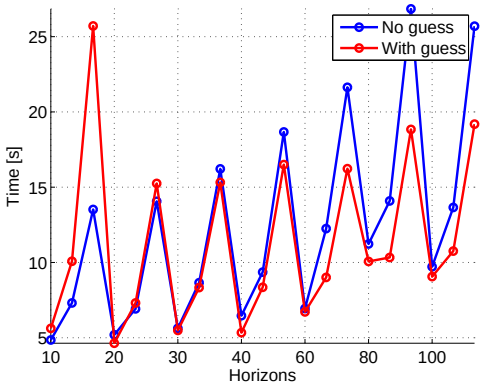
The anomalies appearing in the performance variables are due to spikes in the rates initial guesses, which cause the solver to loose track completely, as happens for cases $N = 10, 20$ of Sh500g. This phenomenon can happen for all the simulation steps, or only for a few, as seen in Sh500g with $N = 50$.

Figure 5.10 shows plots of the solutions for a few different parameters. Con-

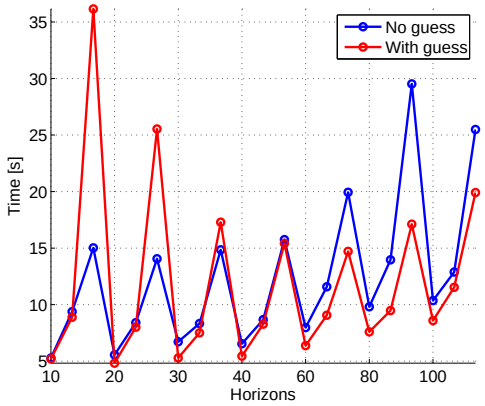


(a) St0-St0g

(b) St100-St100g



(c) St155-St155g



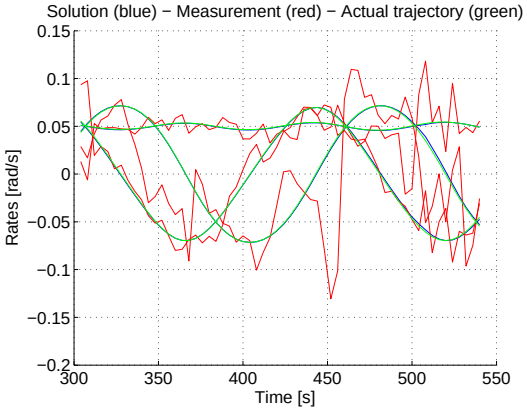
(d) St210-St210g

Figure 5.9: Sorted sequence plots comparing each simulation variations

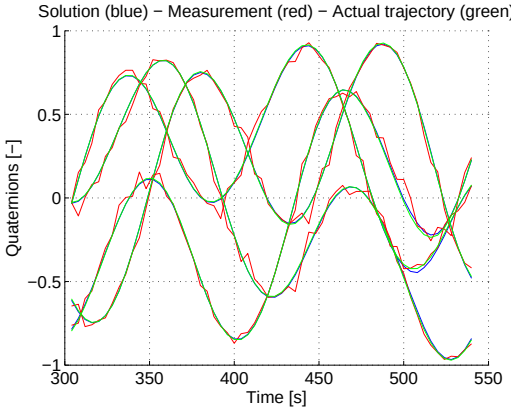
Sim	N	ε_{max}	$\hat{\varepsilon}$	T_{max} [s]	$\hat{T}$ [s]
Sh200g	10	0.1265	0.0415	10.64	7.48
	20	1.3759	0.0643	24.16	12.47
	50	0.0391	0.0094	18.36	16.93
Sh500g	10	2.1267	0.9255	34.24	19.29
	20	1.3952	0.1944	46.03	33.84
	50	1.6444	0.0630	84.97	46.19

Table 5.4: Large horizon simulations performance

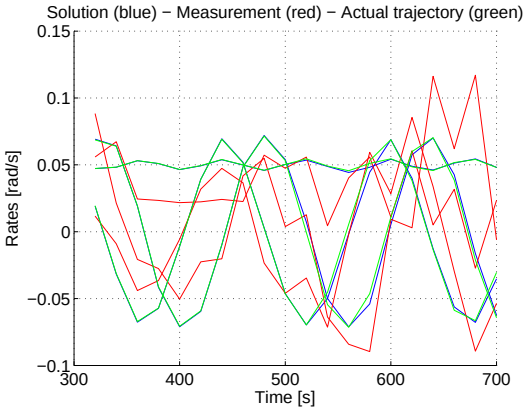
sidering these, one can see the strength of this estimation method, which performs quite effectively even with extremely large sample rates and horizon lengths. Furthermore, improving the robustness is only a matter of using a better differentiation approach, being more adaptive.



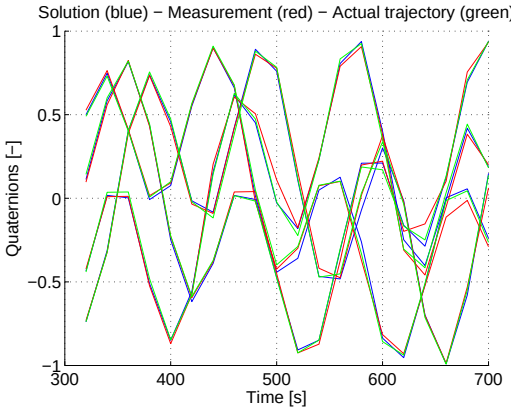
(a) Sh200g rates. $N = 50$



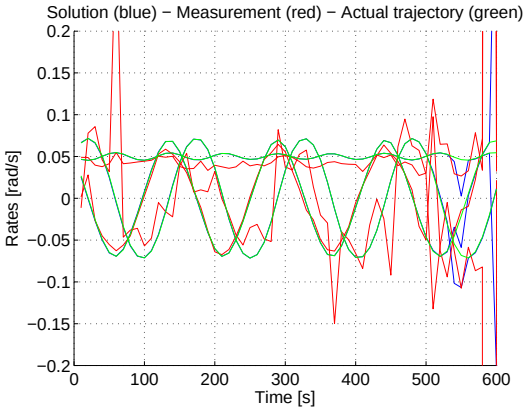
(b) Sh200g quaternions. $N = 50$



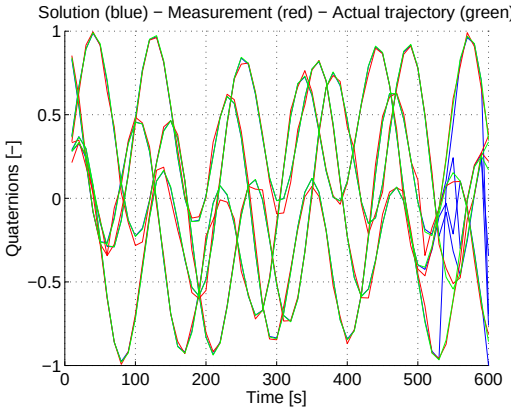
(c) Sh200g rates. $N = 10$



(d) Sh200g quaternions. $N = 10$



(e) Sh500g rates. $N = 50$



(f) Sh500g quaternions. $N = 50$

Figure 5.10: Large horizon simulation solutions

Chapter 6

Final remarks

The high complexity and algorithmic nature of nonlinear optimization hinder any attempts to bound the performance variables of this problem. Thus, an explicit relation between them and the simulation parameters has not been found, should be even possible. On the same grounds, valid conclusion about those relations might only be hinted by performing stochastic problem-modelling experiments such as the Monte Carlo methods. This however, would require high computing resources and a long time period for the analysis, both of which were not reasonable under the scope of the present thesis. Even though, a similar analysis but in a smaller scale has been performed, the results of which have been shown throughout this document, and have demonstrated reasonable and understandable conclusions.

One of the performance variables of interest, the solution error, unfortunately shows quite random behaviour as horizon length and sample rate vary. Even with this randomness however, a tendency of an increasing accuracy for smaller sample rates has actually been observed. On the other hand, the relation with the horizon length is very blurry, although larger horizons appear to perform worse, in terms of both error and computation time. It is important to note as well, that the accuracy obtained has always been quite high, even for the worst cases, which is a significant advantage to current estimation techniques such as Kalman filtering, which do not perform well in the highly nonlinear conditions of the present application.

Another very important performance indicator, perhaps the main one, is the computation time or duration of the estimation algorithm. The reason for this, stems from the fact that on final application for which this approach is targeted, the estimation problem should on-line under real-time conditions. Thus the importance of a bound for the computation time.

As extracted from the plots, a strong tendency has been observed between the computation time and the simulation parameters. On the one hand, it appears to increase linearly with the horizon length. On the other hand, there seems to be a quadratic relation with the sample rate, showing increasing computation time as

the sample rate decreases. Moreover, the validity of these conclusions is supported by the similar tendency, or shape, displayed in the comparison plot (Figure 5.9).

Adding to the significant advantage that this method's accuracy provides, it is also capable of performing well for a very wide range of horizon lengths and sample rates. More specifically, these have spanned values from 10 to 500 seconds of horizon length, and from 0.2 to 50 seconds of sample rate. Note however that only combinations of these yielding sample's number between 10 and 100 have been analysed, since different combinations are unreasonable due to the insignificant performance increase that the provide, if any at all.

Further improvements can be applied to the current algorithm in order to improve its performance, such as using an estimation of the rates to be fed into the optimizer. As seen before, this technique tends to decrease the computation time need by the solver, as well as adding the capability of employing very large horizon lengths that would not be possible otherwise.

In summary, the present approach shows good accuracy and a wide working range, very suitable for the application that is ultimately intended for. At this moment however, special improvements must be introduced to reduce the computation time as well as to find a bound for it, in order to be able to fit in an on-line real-time environment. Future research is then encouraged as this estimation method shows good promise, in special where other methods present serious difficulties.

Bibliography

[1] Shakil Ahmed. Satellite attitude estimation and control using the earth magnetic field. Technical report, Imperial College London, 2010.

[2] John T. Betts. *Practical methods for optimal control using nonlinear programming*. Society for Industrial and Applied Mathematics, 2001.

[3] COIN-OR. Interior point optimization solver IPOPT software package. https://projects.coin-or.org/Ipopt.

[4] CubeSat.org. CubeSat satellites. http://www.cubesat.org, 2008.

[5] Moritz Diehl, Hans Georg Bock, and Johannes P. Schloder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control Optim. Vol. 43, No. 5, pp. 1714-1736*, 2005.

[6] Paola Falugi, Eric Kerrigan, and Eugene Van Wyk. Imperial college london optimal control software - ICLOCS. http://www.ee.ic.ac.uk/ICLOCS/.

[7] Graham C. Goodwin, Juan I. Yuz, Juan C. Agüero, and Mauricio Cea. Sampling and sampled-data models. In *American Control Conference*, 2010.

[8] Alan C. Hindmarsh and Radu Serban. *User documentation for CVODES v2.6.0*. Sundials, 2009.

[9] Pavel Holoborodko. Smooth noise robust differentiators. http://www.holoborodko.com/pavel/numerical-methods/numerical-derivative/, 2008.

[10] Sundials. ODE solver with sensitivity analysis capabilities. https://computation.llnl.gov/casc/sundials/main.html.

[11] Eugene Van Wyk. Tube-based predictive control. Technical report, Imperial College London, 2007.