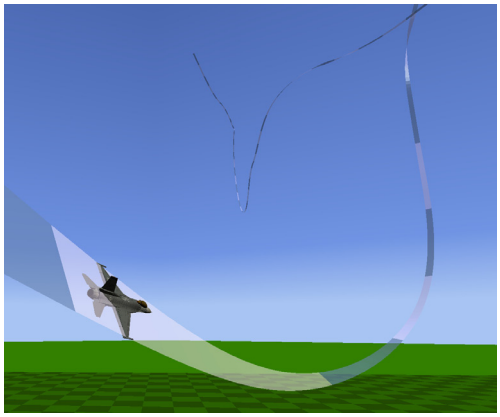
Department of Applied Mechanics

Application for Inverse Simulation of Flight Tracks

Antti Leinonen



Aalto University

SCIENCE + TECHNOLOGY

RESEARCH REPORT

Aalto University publication series
SCIENCE + **TECHNOLOGY** 8/2012

Application for Inverse Simulation of Flight Tracks

Antti Leinonen

Aalto University
School of Engineering
Department of Applied Mechanics
Aeronautical Engineering

Aalto University publication series
SCIENCE + **TECHNOLOGY** 8/2012

ISBN 978-952-60-4541-2 (printed)
ISBN 978-952-60-4542-9 (pdf)
ISSN-L 1799-4896
ISSN 1799-4896 (printed)
ISSN 1799-490X (pdf)

Graphic design: Antti Leinonen
Images: Antti Leinonen

Unigrafia Oy
Helsinki 2012

Finland

Finnish Air Force

NORDIC ECOLABEL
441 697
Printed matter

Author
Antti Leinonen

Name of the publication
Application for Inverse Simulation of Flight Tracks

Publisher School of Engineering

Unit Department of Applied Mechanics

Series Aalto University publication series SCIENCE + TECHNOLOGY 8/2012

Field of research Aeronautical Engineering

Abstract

An application for inverse simulation of flight tracks is implemented. Such an application is required for analysis of flight accidents and verification of flight tracks that are generated using optimization methods. The application could also be used to support monitoring of aircraft fatigue life through the flight parameters. The application is required to utilize a simplified aerodynamic model to analyse flight tracks that are given in the form of the aircraft position at successive time steps. Such data from actual flights could be recorded using a GPS device or a radar.

The most recognized inverse simulation methods are reviewed. The method selected for this work is based on successive numerical differentiation of the flight track to obtain velocity and acceleration vectors. To obtain a solution for the flight attitude, coordinated flight is assumed, i.e. the aircraft has no sideslip or side force. The contribution of the thrust to the lift is considered, and a technique is formulated to recognize the sections of the flight track flown using a negative load factor. Also the wind conditions are considered. The application is implemented in the Matlab® environment.

The inverse simulation is verified using data from flight simulations. It is shown that the results of the inverse simulation are accurate if the initial data are accurate and the assumption of coordinated flight is valid. The flight parameters that are computed via the inverse simulation include speed, Mach number, thrust setting, angle of attack, Euler angles, angular rates and load factor vector. Data from actual flights may require processing, i.e. smoothing and filling, before good results can be obtained.

Keywords inverse flight simulation, accident parameter reconstruction

ISBN (printed) 978-952-60-4541-2 **ISBN (pdf)** 978-952-60-4542-9

ISSN-L 1799-4896 **ISSN (printed)** 1799-4896 **ISSN (pdf)** 1799-490X

Location of publisher Espoo **Location of printing** Helsinki **Year** 2012

Pages 71

Tekijä
Antti Leinonen

Julkaisun nimi
Sovellus lentoradan käänteiseen simulointiin

Julkaisija Insinööritieteiden korkeakoulu

Yksikkö Sovelletun mekaniikan laitos

Sarja Aalto University publication series SCIENCE + TECHNOLOGY 8/2012

Tutkimusala Lentotekniikka

Tiivistelmä

Tässä työssä luodaan sovellus lentoratojen käänteiseen simulointiin. Sovellusta tarvitaan lento-onnettomuuksien analysointiin ja optimointimenetelmillä tuotettujen lentoratojen verifiointiin. Sovellusta voidaan käyttää myös lentokoneen väsymisiän valvonnan apuna lentoparametrien tarkastelun kautta. Vaatimuksena on yksinkertaistetun aerodynaamisen mallin käyttäminen sellaisten lentoratojen analysointiin, jotka on annettu lentokoneen paikkatietona ajan funktiona. Kyseisen kaltaista tietoa voidaan tallentaa GPS- tai tutkamittauksin.

Työssä esitellään merkittävimmät käänteissimulaatiomenetelmät. Tähän työhön valittu menetelmä perustuu lentoradan toistuvaan numeeriseen derivointiin nopeus- ja kiihtyvyysvektoreiden laskemiseksi. Lentoasennon ratkaisemiseksi oletetaan, että lentokone lentää koordinoidusti. Toisin sanoen lentokone ei lennä sivuluisussa, ja sivuvoima on nolla. Työntövoiman aiheuttama lisä nostovoimaan otetaan huomioon, ja työssä kehitetään menetelmä, jolla voidaan tunnistaa negatiivisella kuormitusmonikerralla lennetyt lentoradan osat. Myös tuulen vaikutus huomioidaan. Sovellus ohjelmoidaan Matlab®-ympäristössä.

Käänteissimulaation toiminta verifioidaan käyttämällä lentosimulaatioilla tuotettuja lentoratoja. Työssä osoitetaan, että käänteissimulaation tulokset ovat tarkkoja, jos lähtötiedot ovat tarkkoja ja oletus koordinoidusta lennosta pätee. Käänteissimulaatiolla laskettavia lentoparametreja ovat nopeus, Machin luku, työntövoima-asetus, kohtauskulma, Eulerin kulmat, kulmanopeudet ja kuormituskerroinvektori. Todelliselta lennolta tallennettu lentorata saattaa tarvita esikäsittelyä, kuten sileyttämistä ja täydentämistä, jotta käänteissimulaatiolla saavutetaan hyviä tuloksia.

Avainsanat käänteinen lentosimulaatio, käänteissimulaatio, onnettomuustutkinta

ISBN (painettu) 978-952-60-4541-2 **ISBN (pdf)** 978-952-60-4542-9

ISSN-L 1799-4896 **ISSN (painettu)** 1799-4896 **ISSN (pdf)** 1799-490X

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2012

Sivumäärä 71

FOREWORD

This study was carried out in the Aerodynamics Research Group of the Department of Applied Mechanics at Aalto University. The aim was to select a method for inverse flight simulation and to create an application which is used to analyse how an aircraft has flown on a given flight track. The author would like to express gratitude to the Finnish Air Force for the funding of this work and to Timo Sailaranta and Professor Timo Siikonen for offering their guidance and expertise.

Espoo, February 2012
Antti Leinonen

TABLE OF CONTENTS

NOMENCLATURE

Symbols

Vectors, matrices and quaternions (bold):

$\mathbf{a}$	acceleration vector
$\mathbf{B}$	direction cosine matrix
$\mathbf{e}$	vector of errors
$\mathbf{F}$	force vector
$\mathbf{g}$	gravitational acceleration vector
$\mathbf{J}$	Jacobian matrix
$\mathbf{n}_B$	load factor vector
$\mathbf{p}$	aircraft position vector
$\mathbf{q}$	quaternion
$\mathbf{q}_p$	quaternion of the aircraft orientation
$\mathbf{R}$	rotation matrix
$\mathbf{S}_\alpha$	rotation matrix from body to stability axes
$\mathbf{u}$	vector of control commands or rotation axis
$\mathbf{V}$	velocity vector
$\mathbf{x}_B$	unit vector of aircraft body x-axis
$\mathbf{y}$	vector of state variables
$\mathbf{y}_B$	unit vector of aircraft body y-axis
$\mathbf{z}_B$	unit vector of aircraft body z-axis

Greek:

$\mathbf{\Omega}_q$	special matrix of angular velocities
$\boldsymbol{\omega}$	angular velocity vector

Scalars:

a	speed of sound
C_L	lift coefficient
C_D	drag coefficient
C_Y	side force coefficient
d	distance
F	force
g	gravitational acceleration
g_{sign}	sign of load factor
H	height (altitude)
Ma	Mach number

m	mass
n	component of load factor vector
o_{init}	initial orientation of the aircraft
P	angular velocity about body x-axis
q	dynamic pressure
Q	angular velocity about body y-axis
R	angular velocity about body z-axis
R_0	specific gas constant
S	wing area
T	thrust or temperature
t	time
u	x-component of velocity in ABC-frame
V	speed
v	y-component of velocity in ABC-frame
w	z-component of velocity in ABC-frame
W	weight
x	x-coordinate value
y	y-coordinate value
z	z-coordinate value

Greek:

α	angle of attack or temperature drop rate
β	sideslip angle
γ	ratio of specific heats
Δt	time step
θ	Euler angle of pitch
ρ	density
τ_P	roll time constant
τ_Q	pitch time constant
φ	rotation angle
ϕ	Euler angle of bank (roll)
ψ	Euler angle of yaw (heading)

Subscripts:

0	sea level value
a	air
b	bottom
B	aircraft body coordinate frame (ABC)
comm	control command
corrected	corrected value
D	desired or drag
final	final value
g	ground

i	index
init	initial value
inv	result of inverse simulation
j	index
L	lift
N	normal
NED	north-east-down coordinate frame
NEH	north-east-height coordinate frame (left-handed)
neglected	neglected value
new	new value
p	aircraft orientation
q	quaternion
return	correction to cancel a part of a rotation
S	stability axes
T	thrust or tangential
w	wind
x	*x*-direction
Y	side force
y	*y*-direction
z	*z*-direction

Greek:

α	angle of attack
β	sideslip angle

Abbreviations

5DOF	five degrees of freedom
ABC	aircraft body coordinate frame
cg	center of gravity
GPS	Global Positioning System
ISA	International Standard Atmosphere
NED	north-east-down coordinate frame
NEH	north-east-height coordinate frame (left-handed)

1 INTRODUCTION

1.1 BACKGROUND AND OBJECTIVE

In aircraft flight mechanics, inverse simulation is commonly conceived as computing the control commands which produce a predefined flight path. As the term suggests, it is an inverse technique compared to normal flight simulation, where the flight path is computed from control input. Inverse simulation can also be conceived as computing flight parameters other than the actual control commands, e.g. Euler angles, angular rates, angle of attack and thrust force. These parameters can also be considered control commands in the sense that they directly control the flight.

Inverse simulation techniques have been used for a number of different analyses in flight dynamics. In a design process they can be used to estimate aircraft handling qualities by evaluating the pilot control effort required to fly a predefined track. In fact, the majority of the scientific papers on inverse flight simulation seem to concern the controllability of helicopters. Other applications can be found in the design of digital flight control systems, verification of optimized flight tracks, fatigue life estimation and aircraft accident simulations.

In the Aerodynamics Research Group, some work has already been done in the field of inverse simulation. Inverse simulation has been used for fatigue life management systems [1,2] and for verification of optimized flight tracks for missile avoidance [3,4]. Also, the flight of a missile has been studied via inverse simulation [5]. This work has been commissioned by the Finnish Air Force and the motivation stems from a need for an application for simplified inverse simulation. Such an application is required for analysis of flight accidents and to assess the feasibility of flight tracks that are generated using optimization methods. The application could also be used to support monitoring of aircraft fatigue life.

The application would be used in cases where only a simplified aerodynamic model of the aircraft is available and only the coordinates of the flight path are known. This type of flight path data can be recorded using a GPS device or a radar. The flight parameters which should be computed via the inverse simulation include speed, orientation (pitch, bank, and heading), angle of attack, thrust force, angular velocities and load factors. Due to limitations of the simple aerodynamic model, the actual control

commands or control surface deflections cannot be computed. Computation of control surface deflections would require a detailed model of the aerodynamics and the flight control system, which are not available for all aircraft. However, the angular velocities and the thrust force can be considered as the effective control commands.

In inverse simulation problems, the input is a known flight path as a function of time. The flight path can be given in a number of ways, e.g. the accelerations and angular velocities of the aircraft, or simply the coordinates of the aircraft over time. The objective of this work is to analyse data which is given in the latter form, i.e. only the coordinates $[x(t)\ y(t)\ z(t)]^T$ are known. This poses a problem because the aircraft could have flown the same track in more than one way. A simple example of this is given by considering a straight and level flight path. The aircraft could fly through the path in a wings level orientation, or in a sideslip with a bank angle. Some aircraft could even fly through the path in an inverted flight. Therefore, there will be no unique solution to the problem unless an additional constraint is introduced. A natural choice is to assume that the aircraft flies coordinated flight, i.e. there is no sideslip. This assumption is justified in most cases, because coordinated flight is the normal way to fly an aircraft through most manoeuvres, thus, the sideslip angle is normally zero or small throughout the flight. This assumption is maintained throughout this work, although it will be shown that the algorithm for inverse simulation can detect the points of the track that could not have been flown without sideslip.

1.2 PROJECT DESCRIPTION

A number of different algorithms have been developed for inverse simulation, and a short review of the existing algorithms will be presented. A choice is made that best serves the objective of this work, but none of the existing algorithms were used as is. Instead, the author formulated an approach through straight forward computations. The principle of the approach resembles the early work of Bach and Wingrove [6].

To verify the algorithm for the inverse simulation, firstly, a normal forward flight simulation is implemented. The flight simulator is used to create flight tracks which will be used as test inputs to the inverse simulation. The flight simulator and the inverse simulator use the same simplified aerodynamic model of the aircraft. A model of the F-16 is used. The assumption of

coordinated flight will be built into the flight simulator, because the same assumption will be used in the inverse simulation. More specifically, in the flight simulation the aircraft automatically maintains zero sideslip angle. As this degree of freedom is removed, the simulator will be called a 5DOF simulator. It is then shown that the inverse simulator can reconstruct the flight parameters exactly using only the flight track $[x(t)\ y(t)\ z(t)]^T$, the wind conditions and the aerodynamic model as the inputs.

After verification, the inverse simulation is tested on data that simulates an actual flight. This data are created by manually flying a high fidelity flight simulator, Hutfly2, which is available at the Aerodynamics Research Group [7]. Hutfly2 accurately simulates the aerodynamics and flight control system of the F-16. As a natural consequence, in comparison to the 5DOF simulator, unintended sideslip is occasionally present. Inverse simulation of Hutfly2 flight tracks show that the assumption of zero sideslip is reasonable: The results are good if the pilot does not intentionally induce sideslip.
It is acknowledged that the data created by Hutfly2 does not have any noise or anomalies which would be present in actual GPS or radar recordings. Actual data would require processing, i.e. smoothing and/or filling. The Hutfly2 data therefore simulates an ideal situation where the processing of the data would have been done perfectly.

In the next chapter, an overview of the inverse simulation methods is presented. Chapter 3 describes the 5DOF simulator along with conventions and assumptions that are used in the simulations of this work. The algorithm of the inverse simulation is described in Chapter 4, including the rationale for choosing the algorithm. Test results of the inverse simulation are presented in Chapter 5 and the features of the application are explained in Chapter 6. Finally, a summary is given in Chapter 7.

2 INVERSE SIMULATION METHODS

A brief history of the development of inverse simulation and a review of the current principles have been presented by Thomson and Bradley in 2006 [8]. Attempts were made as early as the 1930s, but only the advent of low-cost computers in the late 1970s and early 1980s made it possible to study complex problems. The possibility of automatic control via inverse simulation methods has been one of the principal motivators for research, as well as the assessment of helicopter handling qualities. [8]

The common objective of most methods is to find the control input, including the thrust setting, but in some cases the objective is to find solely the aircraft motion. Such applications are usually related to analysis of aircraft accidents and place less requirements on the aircraft model. An overview of the most recognized generally applicable methods is presented in chronological order. The rationale for choosing an algorithm for this work is explained in Chapter 4.

2.1 METHOD OF BACH AND WINGROVE

Although not mentioned in the presentation by Thomson and Bradley, the work of Bach and Wingrove can be considered an early implementation of inverse simulation. In their report of 1980 [6], they describe the method used in the Ames Research Center to assist in the analysis of aircraft accidents. They derive equations to determine the aircraft motion from two different types of data: Air Traffic Control radar data and onboard foil recorder data. The format of the radar data is the time history of the aircraft position, while the foil data include indicated airspeed, magnetic heading, barometric altitude and normal acceleration. To obtain a solution, the side force and the sideslip angle are assumed to be negligible.

In the case of the radar data, their algorithm proceeds as follows: Data are smoothed and estimates for the velocity $[\dot{x}\ \dot{y}\ \dot{x}]^T$ and acceleration $[\ddot{x}\ \ddot{y}\ \ddot{z}]^T$ are obtained via differentiation. The estimated wind velocity is subtracted from the velocity vector to obtain the airspeed vector. The gravitational acceleration is subtracted from the acceleration vector to obtain the required aerodynamic and thrust forces. The wind axis Euler angles are then calculated using trigonometry. To obtain a solution for the wind axis bank angle, the flight is assumed to be coordinated, i.e. the sideslip and side force

are assumed to be negligible. The angle of attack is then estimated using the required lift force and aerodynamic data of the aircraft, after which the body axis Euler angles can be calculated. The results provide a history of the aircraft motion to assist in the analysis of aircraft accidents.

2.2 DIFFERENTIATION METHOD

A general algorithm to compute the actual required control commands to fly a predefined track was developed in 1986 by Kato and Sugiura [9] for aircraft, and by Thomson and Bradley [10] for helicopters. The algorithm is referred to as the differentiation method, and, like the algorithm of Bach and Wingrove, is based on numerical differentiation of a discretized flight path. The predefined flight path should be described with enough constraints to allow for a unique solution to be found. The assumption of coordinated flight, also used by Bach and Wingrove, is discussed as one possibility. The angular rates are obtained from the flight path and, introducing dimensional stability derivatives, a set of nonlinear algebraic equations is formed from the equations of motion. The control commands are then solved from the equations in an iterative manner at each time step.

The differentiation method has been applied successfully to some practical problems, and it is considered the first general algorithm for the computation of the control commands. On the downside, the method is complicated and the mathematical model of the aircraft is embedded in the set of the equations. Therefore, making changes to the problem or the aerodynamic model may require rewriting of the algorithm.

2.3 INTEGRATION METHOD

The integration method was developed by Hess, Gao and Wang in 1990 [11]. In comparison with the differentiation method, the integration method is more flexible and suitable for general applications. The integration method utilizes a normal forward flight simulation as a part of the algorithm. The prescribed flight track is discretized using a time step that is typically an order of magnitude larger than the integration step of the forward simulation. At each time step, the previous values of the control commands are taken as the initial guess, and the flight is simulated to the next time step. The error between the actual output and the desired output is computed, and if the error is too large, the controls are corrected and the

interval is simulated again. The iteration at each time step is continued until the error is sufficiently small.

To compute the new controls at each iteration, Newton's method is used. This requires a Jacobian matrix containing the derivative of the output with respect to each control. The elements of the Jacobian matrix are determined by perturbing each control individually and simulating the flight over the interval to obtain the respective change in the output. Once the Jacobian matrix has been determined, the iteration step using Newton's method is

$$\mathbf{u}_{i+1} = \mathbf{u}_i - \mathbf{J}^{-1}\mathbf{e} \tag{2.1}$$

where **u** is the vector containing the controls, **J** is the Jacobian matrix and **e** is the error

$$\mathbf{e} = \mathbf{y} - \mathbf{y}_D \tag{2.2}$$

where **y** is the output obtained using the control vector of the previous iteration, and $\mathbf{y}_D$ is the desired output.

In comparison with the differentiation method, the integration method is considered better in terms of solution accuracy and sensitivity. The method is also independent of the type of the vehicle being simulated. The integration method has been used successfully in the Aerodynamics Research Group for validation of optimized flight tracks [3,4] and for a fatigue life management system [1]. According to the related documents, in order to achieve good results, the state variables of the desired output $\mathbf{y}_D$ should be selected to match the controls **u**. For example, if the controls are aileron, elevator, rudder and thrust, an example of a good combination of the state variables would be roll rate, normal acceleration, yaw rate and speed, respectively.

2.4 OTHER METHODS

Besides the above-mentioned techniques there are still some other methods, many of which are modifications of the integration method. One such algorithm utilizes local optimization and was presented by de Matteis et al. in 1994 [12]. It aims to avoid some problems that arise in redundant cases where the number of control inputs exceeds that of the output parameters. Another method was suggested by Avanzini and de Matteis in 1999 [13],

who divided the computation into two phases of different time scales. In the first phase, a coarse time scale is used to compute the time history of attitude angles and the thrust, assuming that the forces resulting from the control movements and angular velocities can be neglected. In the second phase, a finer time scale is used to compute the control commands that are required to generate the angular velocities obtained from the first phase. The integration method can be used in both phases. The benefits of the two-time-scale method are faster computational time and elimination of oscillation that may be present in the solution of the controls when using the basic integration method. The downside is a slight reduction in accuracy.

In the Aerodynamics Research Group, a technique to fit an arc to five consecutive data points has also been utilized. The flight parameters of the middle point are computed from the curvature of the arc, and the results include angle of attack, Mach number, load factor and Euler angles [2]. The technique has been studied as a part of a fatigue life estimation project, but it is not extensively documented.

3 FLIGHT SIMULATION

To test and verify the inverse simulator, some flight tracks and the flight parameters are required. A normal forward flight simulator is implemented for the purpose of generating such data. In the flight simulator, the same assumptions and aerodynamic model will be used as in the inverse simulation. Therefore, when testing the inverse simulation, the flight parameters computed by the inverse simulator should match exactly the original output of the flight simulator. As described in Section 1.2, the assumption of coordinated flight will be used and therefore the simulator will essentially have five degrees of freedom, thus it will be called a 5DOF simulator.

A number of flight simulators have been implemented in the Aerodynamics Research Group and the most advanced and well-documented of these is Hutfly2. The documentation of the Hutfly2 simulator [7] was taken as the starting point for a simplified flight simulator. Hutfly2 has been implemented in Matlab utilizing the Simulink interface, but for a simplified flight simulation, an implementation using normal Matlab functions is more straightforward. For consistency, the conventions and equations of motion closely follow the documentation of Hutfly2, which follows that of the references [14] and [15]. A number of simplifications will be made in the 5DOF simulator, as well as a routine to maintain zero sideslip angle.

3.1 COORDINATE SYSTEMS

The so-called flat-Earth assumption will be used in the simulations of this work, i.e. the curvature of the Earth and the corresponding directional change of the gravitational acceleration vector $\mathbf{g}_0$ will be neglected, as well as the rotation of the Earth. These assumptions are adequate as long as the flight takes place in a relatively small area and within normal aircraft flight speeds. The equations of this section are based on the reference [14].

A number of coordinate frames will be used in the flight simulations. Within the computations, the position of the aircraft is expressed in an Earth-fixed north-east-down (NED) frame as

$$\mathbf{p}_{NED} = \begin{bmatrix} x_{NED} \\ y_{NED} \\ z_{NED} \end{bmatrix} = \begin{bmatrix} x_{NED} \\ y_{NED} \\ -H \end{bmatrix} \tag{3.1}$$

(See Figure 3.1). While the NED frame is used internally in the computations, a more intuitive but left-handed north-east-height (NEH) frame will be used for the input and output format of the aircraft position data.

The equations of motion are expressed in the aircraft body coordinate (ABC) frame, which is fixed to the aircraft body and centered at the aircraft cg. The normal aerospace convention for the x, y and z axes is adopted and the axes point forward, right and down, respectively.

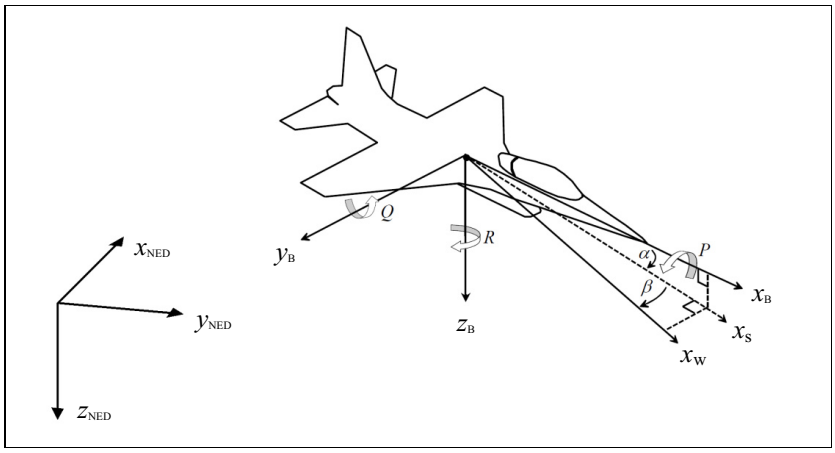


Figure 3.1 Definition of coordinate frames, angular velocities, angle of attack and sideslip angle. [7,14]

The ground velocity vector and the air speed vector of the aircraft are expressed in the ABC frame as

$$\mathbf{V}_g = \begin{bmatrix} u_g \\ v_g \\ w_g \end{bmatrix} \text{ and } \mathbf{V}_a = \begin{bmatrix} u_a \\ v_a \\ w_a \end{bmatrix} \tag{3.2}$$

respectively. The rotational motion of the aircraft is expressed in the ABC frame as the angular velocity vector

$$\boldsymbol{\omega} = \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{3.3}$$

(See Figure 3.1). To determine the motion of the aircraft in the flight simulation, all forces and moments are eventually summed and expressed in the ABC frame.

To express the orientation of the ABC frame with respect to the NED frame, Euler angles or a quaternion can be used. The Euler angles provide an intuitive way to describe the orientation through a sequence of three rotations, whereas the quaternion is more flexible considering the math involved in flight simulations. Euler angles are defined as a sequence of three rotations beginning from the orientation where the aircraft body axes coincide with the north-east-down coordinate axes (Figure 3.2). The positive directions of the three rotations are defined by the right hand rule as follows: The first rotation is about the body z-axis (ψ or yaw/heading) and turns the nose of the aircraft to the right. The second rotation is about the new body y-axis (θ or pitch) and raises the nose. The third rotation is about the new body x-axis (ϕ or roll/bank) and rolls the aircraft to the right. The set of Euler angles is often expressed in the reverse order compared with the order of the rotations. The following range of values is used for the Euler angles in this work:

$$\begin{aligned} -180° < \phi \leq 180° \\ -90° < \theta \leq 90° \\ -180° < \psi \leq 180° \end{aligned} \tag{3.4}$$

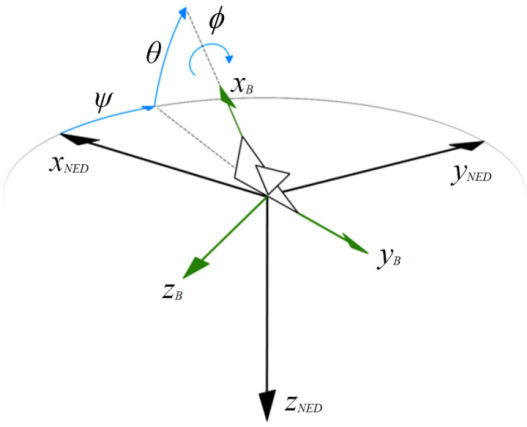


Figure 3.2 Euler angles.

The rotation matrix from the NED-frame to the ABC frame for given Euler angles is the direction cosine matrix

$$\mathbf{B}=\begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix} \tag{3.5}$$

The quaternion that represents the same rotation is related to the Euler angles by

$$\mathbf{q}_p=\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}=\begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2)+\sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2)-\cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2)+\sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2)-\sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \tag{3.6}$$

The representation of the direction cosine matrix using elements of $\mathbf{q}_p$ is

$$\mathbf{B}=\begin{bmatrix} q_1^2+q_2^2-q_3^2-q_4^2 & 2(q_2q_3+q_1q_4) & 2(q_2q_4-q_1q_3) \\ 2(q_2q_3-q_1q_4) & q_1^2-q_2^2+q_3^2-q_4^2 & 2(q_3q_4+q_1q_2) \\ 2(q_2q_4+q_1q_3) & 2(q_3q_4-q_1q_2) & q_1^2-q_2^2-q_3^2+q_4^2 \end{bmatrix} \tag{3.7}$$

The Euler angles are obtained from the elements of **B** or $\mathbf{q}_p$ via

$$\begin{aligned} \phi &= \text{atan2}(b_{23}, b_{33}) \\ \theta &= -\arcsin(b_{13}) \\ \psi &= \text{atan2}(b_{12}, b_{11}) \end{aligned} \tag{3.8}$$

where b_{ij} is the element in the ith row and jth column of **B**. The function atan2 computes the arctangent from two arguments and therefore places the angle in the correct quadrant. It is found as a built-in function in many programming languages, including Matlab.

Two other coordinate frames that are centered on the aircraft cg are used (Figure 3.1). The stability axes are defined by a rotation about the body y-axis. The angle of attack (α) is the angle of this rotation. The transformation from body to stability axes is

$$\mathbf{S}_\alpha=\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{3.9}$$

The wind axes are defined by a subsequent rotation about the stability z-axis so that the resulting wind x-axis points directly to the airflow. The sideslip angle β is defined as the angle of this latter rotation. If the sideslip angle is zero, the wind axes coincide with the stability axes. This will be the case in the simulations of this work, because the sideslip angle is assumed to be negligible.

3.2 SIMPLIFIED AERODYNAMIC MODEL

The aerodynamic model used in this work is based on a simplified format of aircraft performance data which has been developed in the Aerodynamics Research Group [16]. Essentially, the aerodynamic model is a text file which has tables of data in a simple format. The aircraft selected for this work is the F-16 and the data that are used are listed in Table 3.1. For the purpose of this work, the data for lift coefficient C_L were extrapolated to cover also negative angles of attack, as such data were originally missing from the file.

Table 3.1 The data of the simplified aerodynamic model.

Variables	As a function of			Explanation	Constants	Explanation
C_L	α	&	Ma	Lift coefficient	τ_P	Roll time constant
C_D	C_L	&	Ma	Drag coefficient	τ_Q	Pitch time constant
T_{max}	H	&	Ma	Max thrust	P_{max}	Max rate of roll
T_{min}	H	&	Ma	Min thrust	Q_{max}	Max rate of pitch

To determine the values for the variables listed in Table 3.1 in a given flight condition, a built-in interpolation function of Matlab is used. As an example, Figure 3.3 visualizes the data for C_L using a linear interpolation and a spline interpolation. The spline interpolation is used in this work, because it is assumed that the smooth curves approximate the correct values more accurately between the data points.

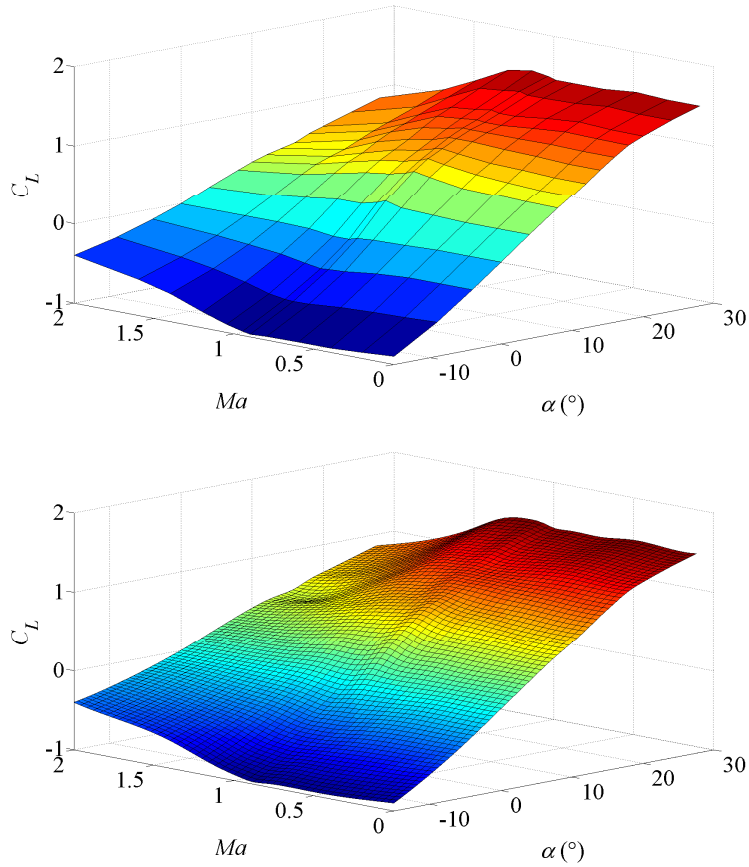


Figure 3.3 C_L as a function of α and Ma. A plot of the linear interpolation on the top and spline interpolation on the bottom.

3.3 ATMOSPHERIC MODEL

A custom function was implemented in Matlab to compute the properties of the atmosphere at a given altitude. The International Standard Atmosphere (ISA) is used as the model of the atmosphere [17,18]. Upon demand, the function can be modified to add optional off-standard atmospheres. The equations of the standard atmosphere are defined separately for the troposphere and the lower part of the stratosphere. The troposphere is defined as the layer between the altitudes of 0 and 11 km, and the equations for the temperature T and density ρ are

$$T = T_0 + \alpha H \tag{3.10}$$

$$\rho = \rho_0 \left(1 + \frac{\alpha H}{T_0}\right)^{-\left(1+\frac{g_0}{\alpha R_0}\right)} \tag{3.11}$$

where

α = -0.0065 K/m	(temperature drop rate with respect to altitude)
T_0 = 288.15 K	(sea level temperature)
ρ_0 = 1.225 kg/m^3	(sea level density)
g_0 = 9.81 m/s^2	(gravitational acceleration)
R_0 = 287.05287 J/kgK	(specific gas constant)

The lower part of the stratosphere is defined as the layer between the altitudes of 11 and 20 km, and the respective equations are

$$T = T_b \tag{3.12}$$

$$\rho = \rho_b e^{\frac{-g_0(H-H_b)}{R_0 T}} \tag{3.13}$$

where the subscript b denotes the following values at the bottom of the layer:

$$T_b = 216.65 \text{ K}$$
$$\rho_b = 0.36391 \text{ kg/m}^3$$
$$H_b = 11000 \text{ m}$$

To compute the Mach number, the local speed of sound is required. It is computed as

$$a = \sqrt{\gamma R_0 T} \tag{3.14}$$

where γ = 1.4 is the ratio of specific heats.

3.4 FLIGHT SIMULATION ALGORITHM

The flight simulation is implemented as a set of Matlab functions. The equations of the flight simulation are based on the documentation of Hutfly2 [7] and its references [14,15], but the author introduced a number of

simplifications as well as a routine to maintain a zero sideslip angle. Prior to running the simulation, the model of the aircraft is read from the text file into the Matlab workspace using a custom function. The simulation time step Δt and the initial values for the state variables $\mathbf{p}_{NED}$, $\mathbf{V}_g$, $\boldsymbol{\omega}$, $\mathbf{q}_p$ must also be defined, as well as a table of the control commands as functions of time. Table 5.1 is an example of such. The simulation then runs in a loop where at each time step the derivatives of the state variables are determined, and the new values are computed using the Euler method for numerical integration. Subsequent to each integration, the sideslip angle is corrected to zero. More specifically, for each time step the following computations are done:

The direction cosine matrix $\mathbf{B}$ is computed from $\mathbf{q}_p$ using (3.7). The wind velocity vector $\mathbf{V}_w$ is determined and transformed into the ABC frame:

$$\mathbf{V}_w = \mathbf{B}\mathbf{V}_{w_{NED}} \tag{3.15}$$

The aircraft true airspeed vector in the ABC frame is obtained by subtracting the wind velocity from the ground velocity:

$$\mathbf{V}_a = \mathbf{V}_g - \mathbf{V}_w \tag{3.16}$$

The airspeed is computed as the length of the velocity vector:

$$V_a = \left|\mathbf{V}_a\right| \tag{3.17}$$

The angle of attack is computed from the components of

$$\mathbf{V}_a = \begin{bmatrix} u_a \\ v_a \\ w_a \end{bmatrix} \tag{3.18}$$

as follows:

$$\alpha = \arctan\left(\frac{w_a}{u_a}\right) \tag{3.19}$$

The density of the air and the speed of sound (ρ and a) in the atmosphere are computed using Eqs. (3.10–3.14). These are then used to compute the dynamic pressure and the Mach number as follows:

$$q = \frac{1}{2}\rho V_a^2 \tag{3.20}$$

$$Ma = \frac{V_a}{a} \tag{3.21}$$

The aerodynamic coefficients C_L, C_D and the thrust force F_T are then interpolated from the tables of the aerodynamic model using the values of α, Ma, H and the thrust setting T_{comm}, which is read from the table of the control commands. A thrust setting of zero corresponds to the minimum thrust value, and a thrust setting of one to the maximum value. The engine response time is neglected, i.e. the thrust force follows the thrust setting without delay. Following from the assumption of zero sideslip, the side force and the value of the aerodynamic side force coefficient C_Y will be zero. The aerodynamic forces in the stability axes can then be obtained as

$$\mathbf{F}_S = qS\begin{bmatrix} -C_D \\ C_Y \\ -C_L \end{bmatrix} \tag{3.22}$$

The sum of all the aerodynamic and thrust forces in the aircraft body frame is then computed by transforming $\mathbf{F}_S$ to the aircraft body axes and adding the thrust force vector $\mathbf{F}_T = [F_T\ 0\ 0]^T$ using

$$\mathbf{F} = \mathbf{S}_\alpha^T \mathbf{F}_S + \mathbf{F}_T \tag{3.23}$$

where $\mathbf{S}_\alpha$ is computed using (3.9).

The time derivatives of the state variables are computed next. The simplified aerodynamic model does not allow the computation of the aerodynamic moments, and for that reason the derivatives of the angular rates are computed on the assumption that the angular rates P and Q follow the commanded rates P_{comm} and Q_{comm} with delays that are determined by the respective time constants τ_P and τ_Q. At this point the derivative of the yaw rate R is set to zero, because the change in the value of R is computed separately following the assumption of no sideslip and no side force. Therefore, the derivative of the angular rate vector is computed as

$$\dot{\boldsymbol{\omega}} = \begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} (P_{comm} - P)/\tau_P \\ (Q_{comm} - Q)/\tau_Q \\ 0 \end{bmatrix} \tag{3.24}$$

The derivative of the aircraft position is obtained by transforming the ground velocity vector to the NED-frame as follows:

$$\dot{\mathbf{p}}_{NED} = \mathbf{B}^T \mathbf{V}_g \tag{3.25}$$

The derivative of the ground velocity vector in the ABC frame is

$$\dot{\mathbf{V}}_g = \frac{1}{m}\mathbf{F} - \boldsymbol{\omega} \times \mathbf{V}_g + \mathbf{B}\mathbf{g}_0 \tag{3.26}$$

where the cross product term takes into account that the rotation of the aircraft changes the velocity components in the ABC frame which rotates with the aircraft body [14].

The derivative of the quaternion $\mathbf{q}_p$ describing the orientation of the aircraft is computed from the current values of $\mathbf{q}_p$ and the angular rates via

$$\dot{\mathbf{q}}_p = -\frac{1}{2}\boldsymbol{\Omega}_q \mathbf{q}_p \tag{3.27}$$

where $\boldsymbol{\Omega}_q$ is a special matrix of the angular rates [14,15]:

$$\boldsymbol{\Omega}_q = \begin{bmatrix} 0 & P & Q & R \\ -P & 0 & -R & Q \\ -Q & R & 0 & -P \\ -R & -Q & P & 0 \end{bmatrix} \tag{3.28}$$

At each time step, the new values of the state variables are then computed using the Euler method of integration. It is acknowledged that Matlab has a number of sophisticated built-in functions for numerical integration, but these can only output an array of the state variables that are integrated. They do not allow a corresponding output of other useful parameters which are computed in the course of the simulation, thus these parameters would have to be recomputed after the simulation. Also, some of the algorithms had trouble dealing

with step inputs that are used for the control for the sake of simplicity. The accuracy of the simulation was evaluated and the results are presented in Subsection 5.1.1. The Euler method proved to be sufficiently accurate for the purpose of creating realistic flight tracks. The integration step is as follows:

$$\begin{bmatrix} \mathbf{p}_{NED} \\ \mathbf{V}_g \\ \boldsymbol{\omega} \\ \mathbf{q}_p \end{bmatrix}_{i+1} = \begin{bmatrix} \mathbf{p}_{NED} \\ \mathbf{V}_g \\ \boldsymbol{\omega} \\ \mathbf{q}_p \end{bmatrix}_i + \Delta t \begin{bmatrix} \dot{\mathbf{p}}_{NED} \\ \dot{\mathbf{V}}_g \\ \dot{\boldsymbol{\omega}} \\ \dot{\mathbf{q}}_p \end{bmatrix}_i \tag{3.29}$$

Because the integration denormalizes the quaternion $\mathbf{q}_p$, it is normalized after the above step [15].

Even though there is no aerodynamic side force, the gravitational force changes the lateral component v of the aircraft velocity if the bank angle is non-zero. This will induce a sideslip angle following the integration (3.29). To implement the assumption of coordinated flight, the orientation of the aircraft will be corrected to a zero sideslip angle at each time step. This can be thought of as an ideal behaviour where the nose of the aircraft always points exactly to the wind in the sideways direction. To accomplish this, the aircraft is rotated the amount of the sideslip angle about its body z-axis. Normally, the sideslip angle is defined as a rotation about the stability z-axis and the value of the sideslip angle is obtained as

$$\beta = \arcsin\left(\frac{v_a}{V_a}\right) \tag{3.30}$$

(Refer to Figure 3.1). However, the correction via a rotation about the body z-axis is chosen instead, because it will lead to a simpler math and will have no effect on the angular rates P and Q as no rotation is done about the body x- or y-axes. The rotation about the body z-axis will change the value of the angle of attack, but this change is negligible because the correction is done at each time step and the required rotation is small. The required correction via the body z-axis is

$$\hat{\beta} = \arctan\left(\frac{v_a}{u_a}\right) \tag{3.31}$$

where v_a and u_a are obtained using (3.16) after the integration (3.29).

The rotation will change some values of the state variables obtained from (3.29). As the orientation of the aircraft is stored in the quaternion $\mathbf{q}_p$, it will be changed. Rotating the aircraft without changing the velocity in the NED frame will also change $\mathbf{V}_g$, because $\mathbf{V}_g$ is the velocity expressed in the ABC frame.

The direction of the body z-axis as expressed in the NED-frame is found by rotating the standard basis vector $\mathbf{k}$ to the aircraft orientation stored in $\mathbf{q}_p$. This is accomplished using quaternion math (See Appendix A) as follows:

$$\mathbf{z} = \mathbf{q}_p \mathbf{k} \mathbf{q}_p^{-1} \tag{3.32}$$

As the orientation of the aircraft is stored in the state variable $\mathbf{q}_p$, the rotation to a zero sideslip angle is accomplished by modifying $\mathbf{q}_p$. This requires another quaternion $\mathbf{q}_\beta$ that represents a rotation of angle $\hat{\beta}$ about the axis $\mathbf{z}$. The quaternion $\mathbf{q}_\beta$ is obtained via (A.6) of Appendix A. The corrected value for $\mathbf{q}_p$ is then obtained by multiplying it by $\mathbf{q}_\beta$:

$$\mathbf{q}_{p_{corrected}} = \mathbf{q}_\beta \mathbf{q}_p \tag{3.33}$$

As the aircraft is rotated about its body z-axis, the components of $\mathbf{V}_a$ lying on the body xy-plane (u_a and v_a) will change. The lateral velocity component v_a will be zero and to maintain the magnitude of the velocity vector on the xy-plane, the value of the longitudinal component u_a is computed using the vector length formula. Hence, the corrected value of $\mathbf{V}_a$ is

$$\mathbf{V}_{a_{corrected}} = \begin{bmatrix} \sqrt{u_a^2 + v_a^2} \\ 0 \\ w_a \end{bmatrix} \tag{3.34}$$

and consequently

$$\mathbf{V}_{g\,corrected} = \mathbf{V}_{a_{corrected}} + \mathbf{V}_w \tag{3.35}$$

Finally, the angular rate R is corrected. In the integration (3.29) the aircraft was rotated about its body z-axis according to the value of the previous time step. The previous value can be thought of as the guess for the required angular rate R to achieve zero sideslip in the current step. The above computation of $\hat{\beta}$ represents the additional correction that is required. To correct the value of R, it is incremented using the value of how much the aircraft was rotated per the length of the time step:

$$R_{corrected} = R + \frac{\hat{\beta}}{\Delta t} \tag{3.36}$$

This concludes the computations which are done for each time step Δt.

Within the simulation, some additional parameters are computed and saved in the output: The Euler angles are obtained from the elements of $\mathbf{q}_p$ using (3.8), and the load factor vector in the aircraft body coordinates is computed as

$$\mathbf{n}_B = \frac{\mathbf{F}}{mg} \tag{3.37}$$

3.4.1 Alternative method for maintaining a zero sideslip angle

Professor Timo Siikonen suggested an alternative and more elegant method to maintain a zero sideslip angle: The sideslip angle could also be maintained at approximately zero by computing the sideslip angle $\hat{\beta}$ using (3.31) and giving the angular rate R the value

$$R = \frac{\hat{\beta}}{\Delta t} \tag{3.38}$$

in Eq. (3.28). This approximately reduces the sideslip angle to zero within the integration (3.29). Although not as accurate, this alternative method could be used to replace Eqs. (3.32–3.36). Differences in accuracy and speed are evaluated in Subsection 5.1.1.

4 INVERSE SIMULATION

To implement an application for the inverse simulation, an algorithm must be selected. The integration method [11] was considered first, as it has been previously used in the Aerodynamics Research Group, and it seems to be the current state of the art. In the integration method, as pointed out in Chapter 2, the desired state variables of the aircraft should correspond to the control variables. The control variables in this case are those used in the 5DOF simulator: P_{comm}, Q_{comm} and T_{comm}. However, initially the aircraft state is given as the time history of its position, and the corresponding variables $[x\ y\ z]^T$ do not match the control variables in the sense that was explained in Chapter 2. A qualifying set of the state variables would be bank angle, normal acceleration and speed for example, but these are among the unknown variables that should be solved by means of the inverse simulation.

Prior to using the integration method, the above-mentioned state variables could be solved using the approach of Bach and Wingrove. This suggests that the problem must be partly solved before the integration method can be used. Therefore, an approach similar to that of Bach and Rodney was selected as the starting point. Moreover, by advancing the computation further, the complete solution is in fact found without the need to use the integration method. The integration method is well suited for finding the time history of the controls, but in this case the controls P_{comm}, Q_{comm} and T_{comm} are not the actual controls by the pilot, and thus need not be computed. Moreover, the simplified aerodynamic model of the aircraft does not allow the computation of the actual controls.

4.1 THE ALGORITHM OF THIS WORK

The principle of Bach and Wingrove [6] to obtain the flight parameter from a given flight track is used. However, rather than deriving symbolic equations through trigonometry, the author formulated a straight-forward approach making use of vector calculus and quaternions. Also, the lift generated by the engine due to angle of attack is considered to obtain the actual thrust of the engine, and Newton's method is utilized to solve the correct combination of the thrust and angle of attack. The author also formulated a technique to determine the sections of the flight track that have been flown using a negative load factor.

Like the method of Bach and Wingrove, the author's algorithm is based on successive differentiation of the flight track to find the required aerodynamic and thrust forces through accelerations. It is assumed that the track data are given as the successive positions of the aircraft at constant time intervals. The time interval Δt should be sufficiently small that the curvature of the track between two consecutive points can be neglected. The data are also assumed to be smooth enough to have continuous first and second derivatives. This would be the case if the track data are an accurate record of an actual flight. However, records of actual flights normally have some noise or other anomalies, by reason of which an appropriate smoothing technique would have to be used prior to attempting the inverse simulation. Ideal test data, which does not require processing, is generated using the 5DOF flight simulation described in Chapter 3 and the Hutfly2 simulator. As explained previously, the assumption of no sideslip and no side force is adapted to enable a unique solution to be found.

Prior to the inverse simulation, the track data are transformed to the NED coordinate frame. The *i*th point of the track data in the NED frame is denoted

$$\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{4.1}$$

For clarity, the index *i* is left out in some of the following equations. In that case all the variables of the equation are of the same index.

The computation of the inverse simulation is begun by numerically differentiating the track data to obtain the velocity and acceleration vectors. The ground velocity is computed as a central difference of three points via

$$\mathbf{V}_{g_i} = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2\Delta t} \tag{4.2}$$

which gives the average velocity over two time steps. The air speed vector is computed by subtracting the estimated wind velocity of the current position and time:

$$\mathbf{V}_{a_i} = \mathbf{V}_{g_i} - \mathbf{V}_{w_i} \tag{4.3}$$

The acceleration is computed as a central difference of three points via

$$\mathbf{a}_i = \frac{\frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\Delta t} - \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{\Delta t}}{\Delta t} = \frac{\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}}{(\Delta t)^2} \tag{4.4}$$

To obtain the total aerodynamic and thrust force, the gravitational acceleration vector is subtracted from the acceleration, and the result is multiplied by the current aircraft mass:

$$\mathbf{F}_i = (\mathbf{a}_i - \mathbf{g}_0) m_i \tag{4.5}$$

The mass of the aircraft can be assumed constant if the fuel consumption within the simulated time period is small. Alternatively, the change in the mass due fuel consumption could be estimated using the solved flight parameters of the previous time step, i.e. the thrust and altitude.

The total force **F** is divided into tangential and normal components. In this work, the normal force is defined as a normal to the air speed vector. The tangential force is defined as a vector projection onto the air speed vector, and the normal force is obtained through a subsequent subtraction:

$$\mathbf{F}_T = \frac{\mathbf{F} \cdot \mathbf{V}_a}{\mathbf{V}_a \cdot \mathbf{V}_a} \mathbf{V}_a \tag{4.6}$$

$$\mathbf{F}_N = \mathbf{F} - \mathbf{F}_T \tag{4.7}$$

Applying the assumption of coordinated flight, the tangential and normal components of the force define a plane that has to be the plane of symmetry of the aircraft (See Figure 4.1). If it were not, there would be a side force component relative to the aircraft body frame. Disregarding the signs, the tangential and normal components of the force are aligned with the stability *x*- and *z*-axes by definition. The scalar stability axis force F_x, including the sign, is obtained via scalar projection as follows:

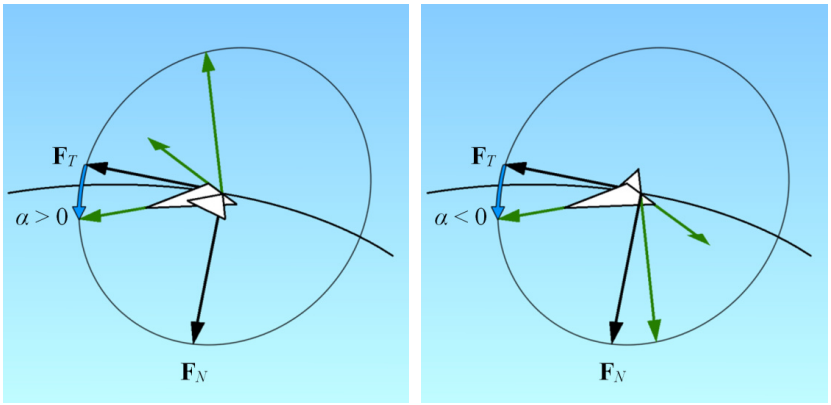
$$F_x = \frac{\mathbf{F} \cdot \mathbf{V}_a}{\|\mathbf{V}_a\|} \tag{4.8}$$

The stability axis force F_z is obtained from the length of the normal force via

$$F_z = -g_{sign} \left\| \mathbf{F}_N \right\| \tag{4.9}$$

where g_{sign} is the sign of the load factor and can have a value of 1 or -1. It denotes whether the aircraft is performing a positive-g or a negative-g manoeuvre. Generally, an aircraft can generate the required normal force in two orientations using either a positive or a negative angle of attack. The two possibilities are illustrated in Figure 4.1. The value of g_{sign} determines the orientation of the aircraft from the two choices: upright or inverted. The sign of the force F_z is opposite to g_{sign}, because a positive load factor corresponds to a negative force in terms of the stability z-axis. The value of g_{sign} is inherited from the previous time step, but it is changed when the aircraft transitions from a positive load factor to a negative load factor or vice versa. A technique to detect such a situation will be discussed later.

The orientation of the aircraft body frame is now obtained by solving the angle of attack simultaneously with the thrust force. Their values are found by requiring that the combined lift, drag and thrust forces generate the stability axis forces F_x and F_z. The lift is defined to be normal to the air speed vector, thus it contributes only to the component F_z. Similarly, the drag contributes only to the component F_x. The thrust force contributes to both components, unless the direction of the thrust force is aligned with the stability x-axis.



a) Positive-g manoeuvre b) Negative-g manoeuvre

Figure 4.1 The normal and tangential forces determine the two possible orientations of the aircraft. Aircraft body axes are shown in green.

To solve the angle of attack and the thrust force, a Matlab function using Newton's method was implemented. The values of the previous time step are used as the initial guess and the iteration step is

$$\begin{bmatrix} \alpha \\ F_T \end{bmatrix}_{j+1} = \begin{bmatrix} \alpha \\ F_T \end{bmatrix}_j - \begin{bmatrix} \frac{\partial F_x}{\partial \alpha} & \frac{\partial F_x}{\partial F_T} \\ \frac{\partial F_z}{\partial \alpha} & \frac{\partial F_z}{\partial F_T} \end{bmatrix}_j^{-1} \begin{bmatrix} F_{x_j} - F_{x_D} \\ F_{z_j} - F_{z_D} \end{bmatrix} \tag{4.10}$$

where F_{x_D} and F_{z_D} are the desired correct values of the forces. The matrix is a Jacobian matrix containing the partial derivatives of the forces with respect to the angle of attack and thrust force. At each iteration step, the derivatives are obtained using finite difference approximation at the current values of the angle of attack and thrust force. The iteration requires repetitive computation of F_x and F_z for different values of the angle of attack and thrust force. They are computed with the equation

$$\mathbf{F}_S = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = qS \begin{bmatrix} -C_D \\ C_Y \\ -C_L \end{bmatrix} + \mathbf{S}_\alpha \begin{bmatrix} F_T \\ 0 \\ 0 \end{bmatrix} \tag{4.11}$$

where the side force coefficient C_Y and the consequently the side force F_y are zero. This follows from the assumption of coordinated flight. In (4.11) the thrust force F_T is assumed to be aligned with the body x-axis. If required, a different alignment could be used as well. The dynamic pressure q is determined using (3.20), and the lift and drag coefficients C_D and C_L are determined as functions of the angle of attack and Mach number from the simplified aircraft model. The built-in spline interpolation function of Matlab is used to determine the values from the tables of aerodynamic data.

If the thrust value obtained from (4.10) is below the minimum thrust value, the difference to the minimum value is given in the results also in the form of an additional drag coefficient acting along the line of thrust. The additional drag may be an indication of airbrake deployment, or in the case of an accident, structural damage.

Having obtained the angle of attack, it is possible to determine the orientation of the aircraft body in the NED-frame. The unit vector of the body y-axis coincides with the stability y-axis, and is obtained using cross product via

$$\mathbf{y}_B = -g_{sign}\frac{\mathbf{F}_N \times \mathbf{V}_a}{\left\|\mathbf{F}_N \times \mathbf{V}_a\right\|} \tag{4.12}$$

where g_{sign} takes into account the correct orientation of the two possibilities (See Figure 4.1). The unit vector of the body x-axis is obtained by normalizing the vector $\mathbf{V}_a$ and rotating it about the body y-axis the amount of angle of attack. The rotation is accomplished using a corresponding quaternion $\mathbf{q}$ via

$$\mathbf{x}_B = \mathbf{q}\left(\frac{\mathbf{V}_a}{\left\|\mathbf{V}_a\right\|}\right)\mathbf{q}^{-1} \tag{4.13}$$

where $\mathbf{q}$ is computed from the required rotation angle and axis using (A.6). The unit vector of the body z-axis is now obtained from the cross product

$$\mathbf{z}_B = \mathbf{x}_B \times \mathbf{y}_B \tag{4.14}$$

As these unit axes of the aircraft body are expressed in the NED-frame, the direction cosine matrix is simply

$$\mathbf{B} = \left[\mathbf{x}_B \,|\, \mathbf{y}_B \,|\, \mathbf{z}_B\right] \tag{4.15}$$

The corresponding quaternion $\mathbf{q}_p$ is obtained from $\mathbf{B}$ using (A.9) and the Euler angles using (3.8). The angular rate vector $\boldsymbol{\omega}$ is obtained next from the quaternions $\mathbf{q}_p$ and its derivative $\dot{\mathbf{q}}_p$. The derivative is computed using the difference of two consecutive values:

$$\dot{\mathbf{q}}_{p_i} = \frac{\mathbf{q}_{p_i} - \mathbf{q}_{p_{i-1}}}{\Delta t} \tag{4.16}$$

Caution is taken when using the above equation, because a quaternion multiplied by -1 represents the same rotation. Therefore, two consecutive quaternions representing slightly different rotations may have different signs, because at certain points, the quaternion returned by (A.9) switches sign. In that case, one of the two quaternions must be multiplied by -1 to get the correct value for the difference in (4.16). To detect the situation of opposite signs, the condition

$$\left\|\mathbf{q}_{p_i} - \mathbf{q}_{p_{i-1}}\right\| > 0.2 \tag{4.17}$$

was found to work well. The value 0.2 is just an empirical choice. The angular rates are now obtained via [14]

$$\boldsymbol{\omega} = \begin{bmatrix} P \\ Q \\ R \end{bmatrix} = \begin{bmatrix} -q_2 & q_1 & q_4 & -q_3 \\ -q_3 & -q_4 & q_1 & q_2 \\ -q_4 & q_3 & -q_2 & q_1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} \tag{4.18}$$

Finally, the load factor vector expressed in the aircraft body frame is

$$\mathbf{n}_B = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{\mathbf{B}^T \mathbf{F}}{mg} \tag{4.19}$$

Note that the definition of the load factor vector and its components differs from the normal definition of the load factor n, which is

$$n = \frac{L}{W} \tag{4.20}$$

4.2 DETECTING NEGATIVE LOAD FACTOR AND LIMITING ROLL RATE

As explained in the previous section, there are two possible orientations of the aircraft to generate the lift force: The normal way using a positive angle of attack and load factor, or an inverted way using a negative angle of attack and load factor. In the case of normal cruising flight, it can be assumed that a negative load factor is not encountered. However, for example in a manoeuvring flight of a fighter, negative load factors can occur. The author formulated the following method to automatically determine the sections of the flight track where the load factor is negative.

At the beginning of the inverse simulation, the correct alternative of the two possible orientations of the aircraft must be known (Figure 4.1). If it is known that initially the aircraft is approximately in an upright orientation, the initial value of g_{sign} can be obtained by examining the initial direction of the normal force vector $\mathbf{F}_N$. If it is pointing upwards, i.e. the z-component is negative, the sign of the load factor has to be positive. This can be formulated into a simple equation

$$g_{sign} = -1 \cdot \text{sign}\left(F_{N_z}\right) \cdot o_{init} \tag{4.21}$$

where o_{init} denotes the initial orientation of the aircraft. Upright orientation corresponds to a value of 1 and inverted to a value of -1. Thus, together with the normal force vector, the initial orientation determines the sign of the load factor at the beginning. Subsequently, the sign (stored in the variable g_{sign}) is preserved, unless it is not physically possible for the aircraft to fly the track without changing the sign.

Figure 4.2 illustrates a typical moment in flight where the sign of the load factor changes as the pilot pushes the stick to induce a rapid nose down manoeuvre. Even though the sideslip is assumed to be zero, the data may represent a flight condition where a small sideslip and side force are present. As a consequence, at the instant the aircraft changes the load factor from positive to negative, the small side force will have a notable effect on the *direction* of the normal force $\mathbf{F}_N$ (Eq. 4.7). This is because the relative magnitude of the side force will be greater at the moment the lift force is close to zero.

The arrows in Figure 4.2a illustrate the directions of the normal force vectors $\mathbf{F}_N$ at six consecutive time steps. In the proximity of the load factor changes from positive to negative, vectors 3 and 4 are diverted to the left because of the small side force component. If the possibility of the sign of the load factor changing were to be ignored, the resulting orientations of the aircraft would be those illustrated in Figure 4.2a. This would result in an unrealistically fast roll rate as the aircraft would have to turn inverted in an instant to continue the manoeuvre using a positive load factor. Therefore, if the angle between two consecutive normal force vectors exceeds 90 degrees, the sign of the load factor is changed. In Figure 4.2a, this happens between points 3 and 4, and the corresponding result of the inverse simulation will be that of Figure 4.2b. As a result, the orientation of the aircraft is not required to change as much, and the aircraft uses a negative angle of attack to generate the normal force from point 4 onwards. Using the threshold of 90 degrees minimizes the required change in the orientation over a single time step. The same method is used to detect the change of the load factor in the other direction, i.e. from negative to positive.

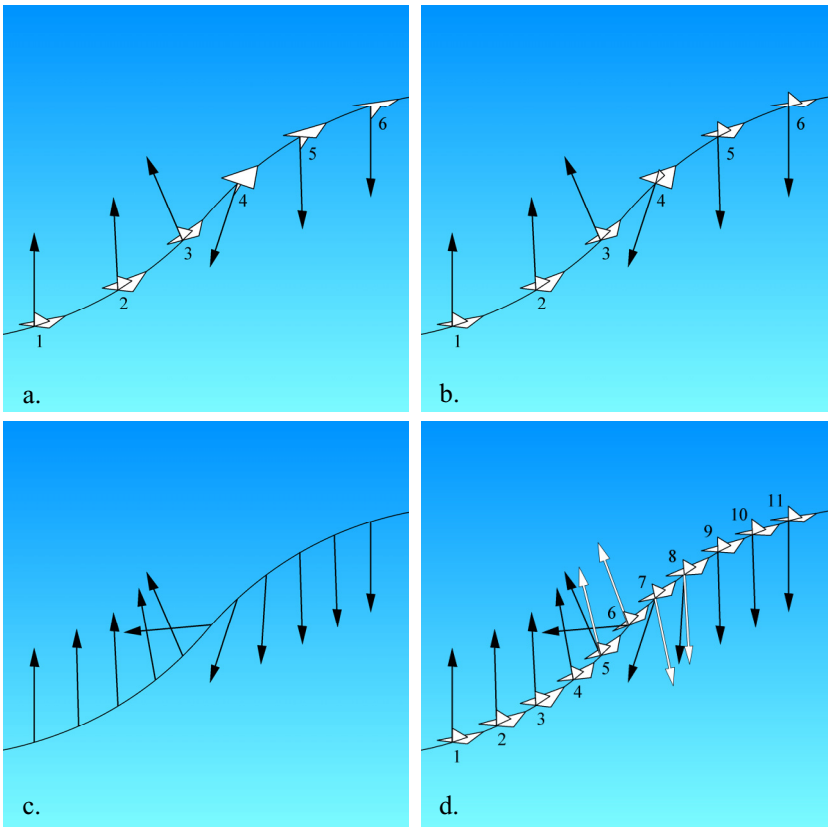


Figure 4.2 Aircraft orientation based on the directions (black) and corrected directions (white) of the normal force vectors. See text for details.

However, the above method, as such, does not work in all cases. If the time step is short, the change in the direction of the normal force vector could be so gradual that there will be no 90-degree jump. Figure 4.2c illustrates this situation. The solution is to limit the roll rate based on the known performance of the aircraft. Figure 4.2d illustrates this: At points 5 and 6 the aircraft is rolled towards the orientation commanded by the normal force vectors (black arrows), but only within the performance parameters of the aircraft. The white arrows denote the actual directions of the normal force that the aircraft can reach when it is tracking the commands of the black arrows. The 90-degree threshold is therefore exceeded at some point, when the next normal force vector (black arrow) is compared to the previous aircraft orientation (white arrow). In Figure 4.2d, this happens between points 6 and 7. The sign of the load factor can thus be changed using the same threshold of 90 degrees. The aircraft now continues the manoeuvre using a negative load factor, while now trying to align the bottom of the aircraft to the normal force vectors.

As is also illustrated in Figure 4.2d, some roll rate is induced in the aircraft as the aircraft attempts to track the diverting normal force vectors. After the sign of the load factor is changed, the aircraft attempts to track the normal force vectors 'the other way' and catches the commanded direction at point 9. As a consequence, there may be a quick wing rock visible in the inverse simulation results at the points where the sign of the load factor changes.

It is also acknowledged that a side force component of the normal force vector must be neglected at points 5–8 of Figure 4.2d. This implies that actually the aircraft must have had some sideslip at those points, because the aircraft could not have rolled fast enough to generate that force using the lift of the wing. The magnitude of the ignored side force coefficient and the respective side load factor are saved to the inverse simulation results. Typically, they are very small and the corresponding sideslip angle would be similarly small, making the assumption of zero sideslip justified.

The aircraft roll performance is limited by imposing a limit to the angular rate P and its rate of change $\dot{P}$. However, limiting $\dot{P}$ may cause roll oscillation right after the sign of the load factor is changed. This is because, as described above, some roll rate may be induced just before the change. Therefore, the limit for $\dot{P}$ is disabled for a duration of 0.5 seconds after the change. This will prevent the oscillation.

4.2.1 The required mathematics

The maximum values of the roll rate and its rate of change are not constant, but depend on the flight condition. The solved flight parameters of the previous time step can be used to determine the current maximum values, or alternatively they can be estimated using constant values. The values are used only to detect the change in the sign of the load factor.

The roll rate P is computed at each time step using Eq. (4.18) and its rate of change is obtained using the difference to the previous value as

$$\dot{P}_i = \frac{P_i - P_{i-1}}{\Delta t} \tag{4.22}$$

If P or $\dot{P}$ is greater than the corresponding maximum value given for the aircraft, the aircraft has rotated excessively about its body x-axis over the last time step. To correct the orientation of the aircraft, the angle of excess rotation is computed first. As is seen in Figure 4.2a, the direction of the

normal force vector can change up to 180 degrees within one time step. Before the correction of the roll rate is done, the orientation of the aircraft changes accordingly. Therefore, a reliable computation of the roll rate is required up to a 180-degree roll within one time step.

It is noted that Eq. (4.18) does not give accurate values for the angular rates if there is a big difference between two consecutive orientations of the aircraft. This is because the derivative obtained from Eq. (4.16) is not accurate for big changes. Also the quaternion used in the matrix of Eq. (4.18) represents only the latter orientation of a time step, which is acceptable only when the two consecutive orientations only have a small difference.

To compute accurate values for the angular rates up to a 180-degree roll per time step, an alternative method is introduced: The rotation axis and angle between two consecutive orientations are determined from the corresponding rotation matrix. As is seen from Eq. (4.15), the direction cosine matrix **B** has the aircraft body axes as the columns. It is therefore easy to see that using a rotation matrix **R** to rotate each body axis, the relationship between two consecutive orientations is

$$\mathbf{B}_i = \mathbf{R}_i \mathbf{B}_{i-1} \tag{4.23}$$

where $\mathbf{R}_i$ is a rotation matrix. Solving $\mathbf{R}_i$ from Eq. (4.23) yields

$$\mathbf{R}_i = \mathbf{B}_i \mathbf{B}_{i-1}^T \tag{4.24}$$

The unit vector **u** of the rotation axis and the rotation angle φ are then solved from **R** using (A.13). The angular rate vector is obtained by multiplying **u** by the rotation angle φ and dividing it by the time step Δt. Expressing the angular rate vector in the aircraft body frame will yield the components P, Q and R. The equation is

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix} = \mathbf{B}^T \mathbf{u} \frac{\varphi}{\Delta t} \tag{4.25}$$

In principle, this equation could replace Eq. (4.18) completely, but the Matlab implementation of algorithm (A.13) was found not to perform well with very small angles. Therefore Eq. (4.18) is used as the default, and if

one of the angular rate components is greater than 5°/s, the rates are recomputed using Eq. (4.25) instead. The reliable computation of angular rates is thus established also for big changes of orientations.

The maximum rate of change of the roll rate P is computed using P_{max} and τ_P as follows:

$$\dot{P}_{\max_i} = \frac{1}{\tau_P}\left(P_{\max} - P_{i-1}\right) \tag{4.26}$$

If the roll rate P_i or its rate of change obtained from (4.22) is too big, its value is corrected accordingly and labelled $P_{i_{new}}$. The aircraft is then rotated back and the required angle of rotation is

$$\varphi_{return} = \left(P_{i_{new}} - P_{i-1}\right)\Delta t \tag{4.27}$$

Even though P is the angular velocity about the aircraft body x-axis, doing the corrective rotation of φ_{return} about the body x-axis would change the angle of attack and induce a sideslip angle if the angle of attack is non-zero. Therefore, the correction is done via a rotation about the stability x-axis and re-computation of the angle of attack. At first the new body (and stability) y-axis is obtained using quaternion math as follows:

$$\mathbf{y}_{B_{new}} = \mathbf{q}\mathbf{y}_B\mathbf{q}^{-1} \tag{4.28}$$

where $\mathbf{q}$ is a quaternion that represents the rotation of φ_{return} about the stability x-axis and is obtained using (A.6). The total force vector $\mathbf{F}$ is then projected onto the new aircraft plane of symmetry. As $\mathbf{y}_B$ is a normal vector to this plane, the projection is accomplished as follows:

$$\mathbf{F}_{new} = \mathbf{F} - \left(\mathbf{F}\cdot\mathbf{y}_B\right)\mathbf{y}_B \tag{4.29}$$

The new total force vector replaces the old one, and the aircraft orientation is recomputed beginning from equation (4.6). The white arrows of Figure 4.2d illustrate the directions of the projected normal force vectors $\mathbf{F}_{new}$. Due the above projection of $\mathbf{F}$, a component of the original $\mathbf{F}$ is neglected. It is a side force component and its magnitude is computed as

$$Y_{neglected} = \mathbf{F}\cdot\mathbf{y}_B \tag{4.30}$$

and the respective side force coefficient is

$$C_{Y_{neglected}} = \frac{Y_{neglected}}{qS} \tag{4.31}$$

The side force coefficient is saved to monitor the error that has been made. Too large values indicate that the assumption of coordinated flight may have produced inaccuracies in the results. The side force component will also be visible in the *y*-component of the body axis load factors, which are computed via (4.19).

5 VERIFICATION AND TESTING

5.1 VERIFICATION USING 5DOF SIMULATOR

To verify the algorithm of the inverse simulation, it is tested on a flight track generated using the 5DOF simulator. As the inverse simulation can also account for the wind, a simple wind vector was defined as a function of the altitude H as

$$\mathbf{V}_{wNED} = \begin{bmatrix} 0 \\ 0.5\sqrt{H} \\ 0 \end{bmatrix} \frac{m}{s} \tag{5.1}$$

This gives a wind velocity vector which always points to the east but whose magnitude varies with altitude. The magnitude at an altitude of 2000 metres is 22.4 m/s.

The flight track consists of a 30-second manoeuvre which begins from straight and level flight at an altitude of 2000 metres. The aerodynamic model of the F-16 is used. The ground velocity is 300 m/s to the north and consequently the nose of the aircraft is pointed slightly to the west because of the side wind. The aircraft then does a half loop, rolls 90 degrees to make a turn to the east and then rolls upright and pushes the nose down to induce a negative load factor. Table 5.1 shows the predefined control commands that were used. For simplicity, the control commands are step inputs and remain constant until the start time of the next line, whereupon new values are specified. The thrust setting is also varied throughout the manoeuvre. The resulting flight track is illustrated in Figure 5.1.

Table 5.1 Control commands of the test manoeuvre.

Start time (s)	P_{comm} (°/s)	Q_{comm} (°/s)	T_{comm} (0..1)
0	0	0	0.8
1	0	14	0.8
3	0	14	1.0
5	0	13	0.9
13	0	13	0.8
15	95	13	0.6
16	0	15	0.5
21	75	6	0.0
22	0	6	0.2
23	0	-6	0.2

5.1.1 Verifying the accuracy of the 5DOF simulator

Prior to attempting the inverse simulation, the accuracy of the 5DOF simulator and the Euler method for integration were evaluated. The evaluation was also done using the alternative method for maintaining a zero sideslip angle, as described in Subsection 3.4.1. It is assumed that small errors in the flight parameters will gradually cumulate to a more visible error in the position of the aircraft as the simulation proceeds. Therefore, the error in the final position of the aircraft d_{final} was taken as the measure of the accuracy. To estimate the significance of the error, it can be compared with the length of the flight track, which is 7.664 km.

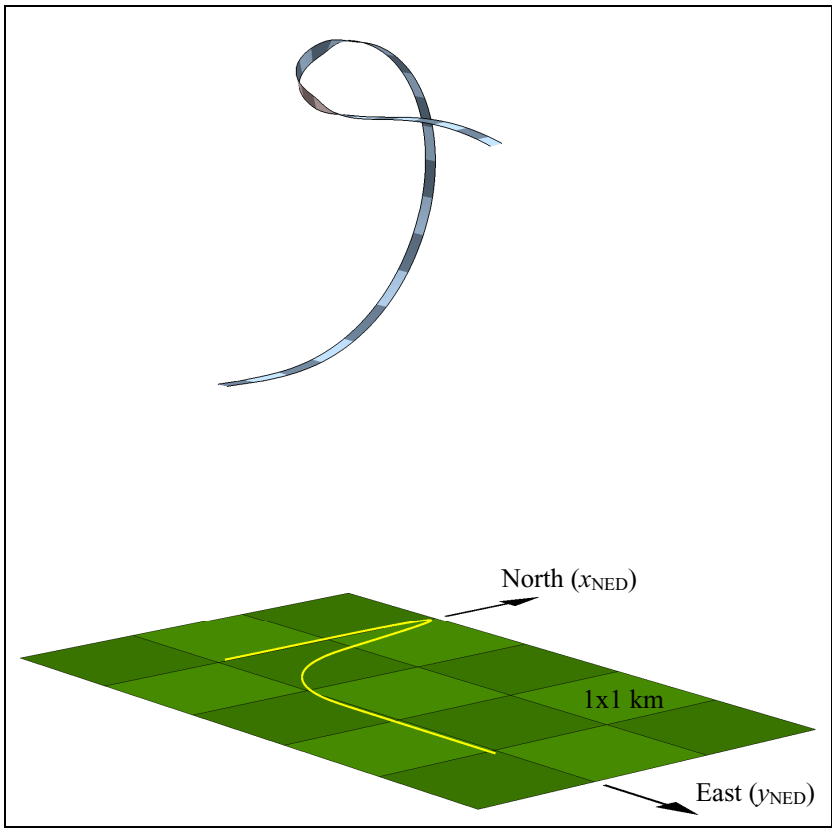


Figure 5.1 The test track created using the 5DOF simulator. One stripe on the ribbon corresponds to one second of flight.

The accuracy was evaluated by running the simulation of the 30-second test track using 10 different integration time steps. The longest time step used was 0.2 seconds. As the time step is gradually decreased down to 0.0002

seconds, the final position of the aircraft seems to converge towards an accurate result. Decreasing the time step further would eventually result in a reduction in the accuracy, as the rounding errors of the floating point computations cumulate. Such divergence is not visible when using the above range of time steps, thus the result obtained using the smallest time step of 0.0002 seconds can be considered the most accurate.

Simulations using both methods for maintaining a zero sideslip angle seem to converge towards the same final position, although the convergence is not monotonic. The alternative method seems slightly less accurate, and therefore, it is assumed that the most accurate result is obtained using the first method and the smallest time step of 0.0002 seconds. The accuracies of the other results are then evaluated by computing the distance of the final position d_{final} to the most accurate result. The results are given in Table 5.2 for the first method, and in Table 5.3 for the alternative method. The improvement in accuracy is also visualized on a logarithmic scale in Figure 5.2, and plots of the paths of convergence are shown in Figure 5.3. The simulations were run using a computer with a 2.66 GHz quad core processor.

Based on the results, the time step of 0.02 seconds is sufficiently fine for the purpose of creating realistic flight tracks. In plots of the flight parameters (such as Figure 5.4), the graphs of 0.02 second time step practically coincide with the more accurate results. The flight track obtained using the first method and the time step of 0.02 seconds was selected to be the test track for the inverse simulation.

Table 5.2 Convergence of the final position of the aircraft towards the most accurate result, as the integration time step is decreased (First method).

Time step (s)	*Frequency* (Hz)	*Computing time* (s)	d_{final} (m)
0.2	5	1.6	65.71
0.1	10	3.1	23.58
0.05	20	6.1	5.20
0.02	50	15.4	7.36
0.01	100	31.6	7.13
0.005	200	66.9	2.87
0.002	500	172.9	0.67
0.001	1000	290.4	0.73
0.0005	2000	582.5	0.44
0.0002	5000	1486.9	0.00

Table 5.3 Convergence of the final position of the aircraft towards the most accurate result, as the integration time step is decreased (Alternative method).

Time step (s)	*Frequency* (Hz)	*Computing time* (s)	d_{final} (m)
0.2	5	1.7	113.96
0.1	10	3.1	28.38
0.05	20	6.1	14.30
0.02	50	14.7	11.76
0.01	100	29.3	9.39
0.005	200	59.0	3.10
0.002	500	146.8	0.41
0.001	1000	295.4	0.96
0.0005	2000	588.1	0.33
0.0002	5000	1485.0	0.07

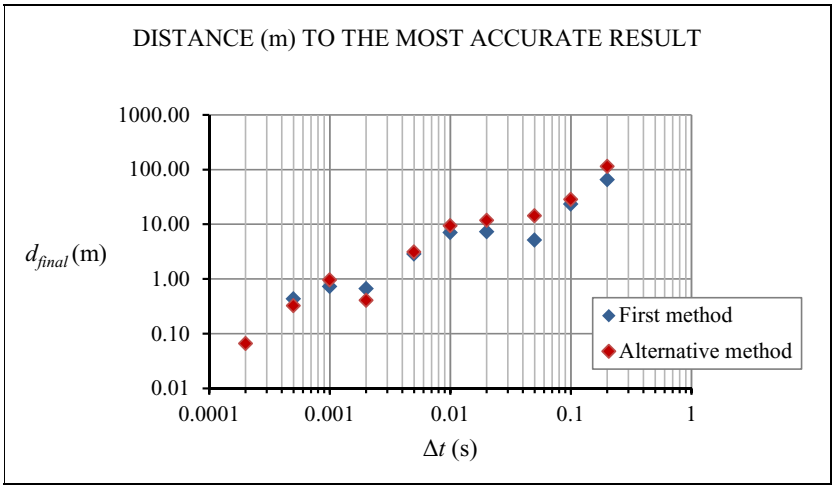


Figure 5.2 Improvement of the 5DOF simulator accuracy with decreasing time step. Results of both methods for maintaining a zero sideslip angle are shown.

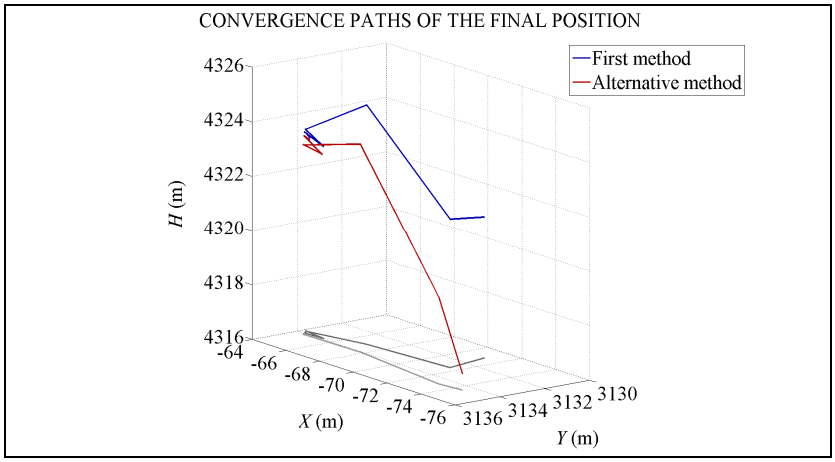


Figure 5.3 Convergence paths of the final position of the aircraft for the time steps from 0.02 to 0.0002 seconds.

5.1.2 Verifying the inverse simulation

To verify the algorithm of the inverse flight simulation, the position data $[x(t)\ y(t)\ z(t)]^T$ of the test track was used as an input to the inverse simulation. The data of the test track were sampled using a time step of 0.2 seconds. Besides the aerodynamic model and the wind data, only the position data were used. The inverse simulation of the 30-second track takes approximately 9 seconds on the above-mentioned computer. The flight parameters resulting from the inverse simulation are compared to the original values of the 5DOF simulation in Figure 5.4. Considering the sparse time step of 0.2 seconds, the inverse simulation accurately captures the angle of attack and the changes in the thrust setting. The transition from positive to negative load factor is also correctly recognized, as is seen from the angle of attack and load factor curves from 26 seconds onwards. The respective plots of the Euler angles and the angular rates are also reconstructed to a good accuracy. The inverse simulation performs as expected and reducing the time step was observed to linearly reduce the errors which are seen in the graphs. Plots of air and ground velocities are not shown, as the values are resolved from the track data by a simple differentiation and are correct by definition.

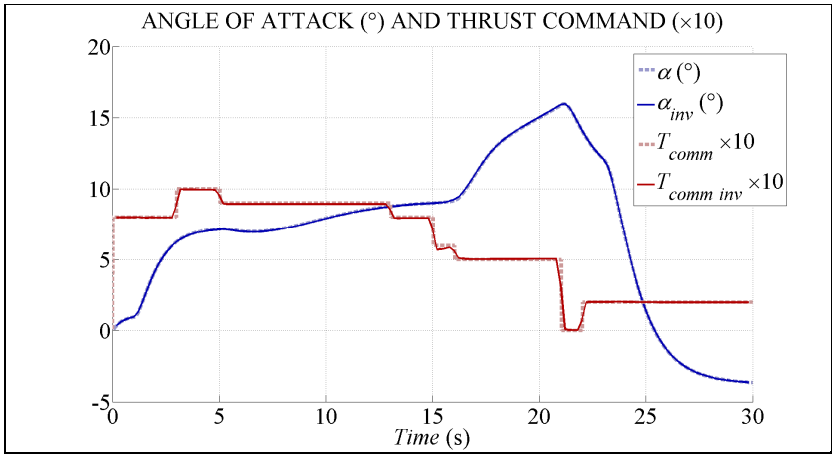


Figure 5.4 Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of the 5DOF simulation.

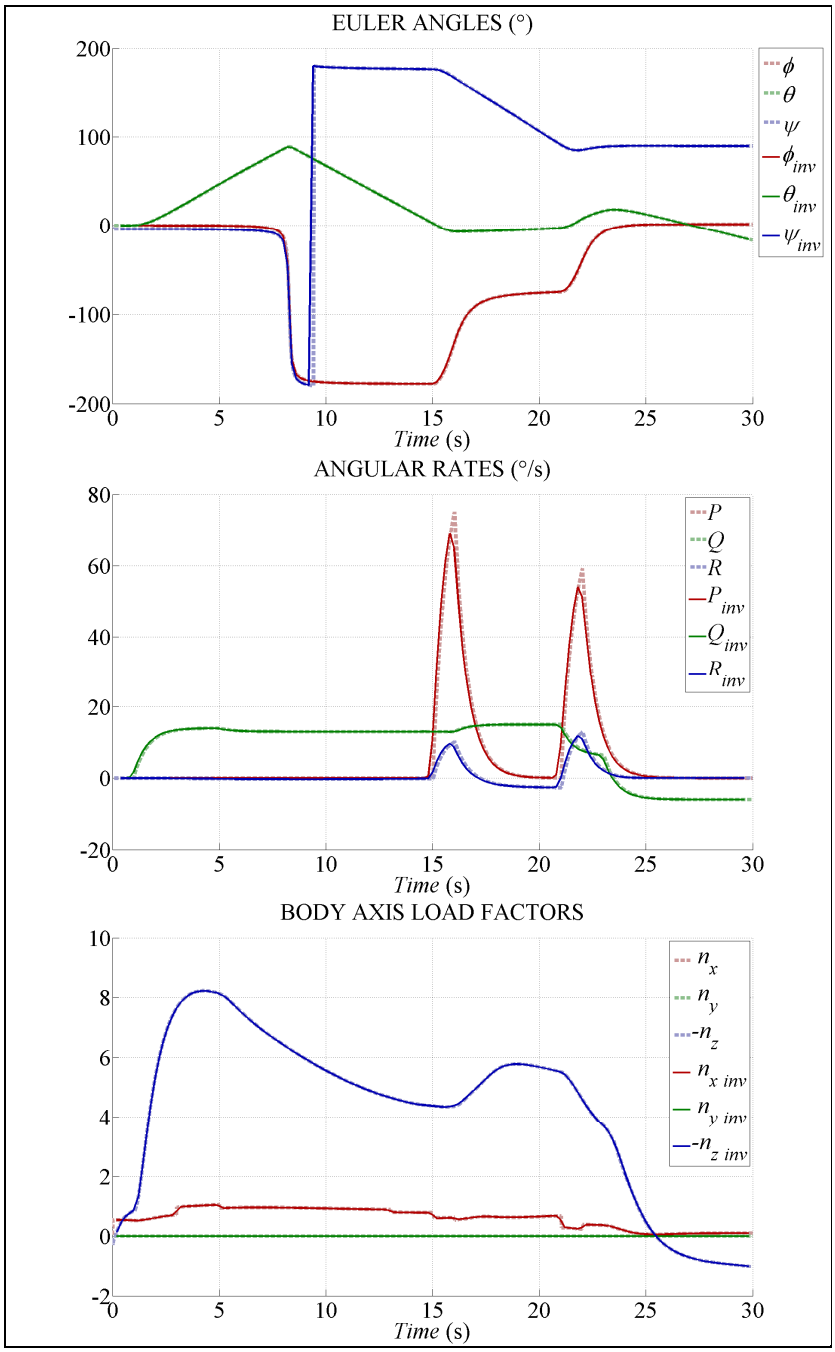


Figure 5.4 (continued) Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of the 5DOF simulation.

5.2 TESTING WITH 6DOF SIMULATOR

The 5DOF simulator accurately realizes the assumption of coordinated flight, so the inverse simulation can perfectly reconstruct the flight parameters. To evaluate the performance in a more realistic case, additional test tracks were generated using the Hutfly2 6DOF simulator via manual joystick control. Hutfly2 includes an accurate simulation of the F-16 aerodynamics and flight control system. Sideslip can occur in the simulation, and the effect of assuming zero sideslip can be studied. A number of simple individual manoeuvres were flown as well as longer irregular tracks. The wind speed was zero in the simulations of these tracks.

5.2.1 Test track number one

The first Hutfly2 test track (Figure 5.5) is similar to the 5DOF test track. Rudder control was not used, allowing the flight control system to minimize the sideslip. The Hutfly2 data was sampled using a frequency of 10 Hz and the inverse simulation was run. The results are compared to the original Hutfly2 data in Figure 5.6. The inverse simulation cannot perfectly track the original values of the angle of attack, and the thrust setting is clearly too small, although the trend of the thrust setting (increase/decrease) is captured well. As the inverse simulation algorithm was found to be accurate, it can be concluded that the inaccuracies follow from differences in the aerodynamic data between Hutfly2 and the simplified aerodynamic model. Some of the errors may also follow from the simplification itself, e.g. negligence of control surface forces. Also, the way the thrust setting is mapped to the actual thrust force may be different. The error of the thrust setting is bigger at smaller angles of attack, from which it may be concluded that the drag coefficient of the simplified aerodynamic model is too small, especially at smaller angles of attack. In fact, the graph can be considered a representation of a situation where an accurate aerodynamic model cannot be constructed, but an estimate has to be used instead.

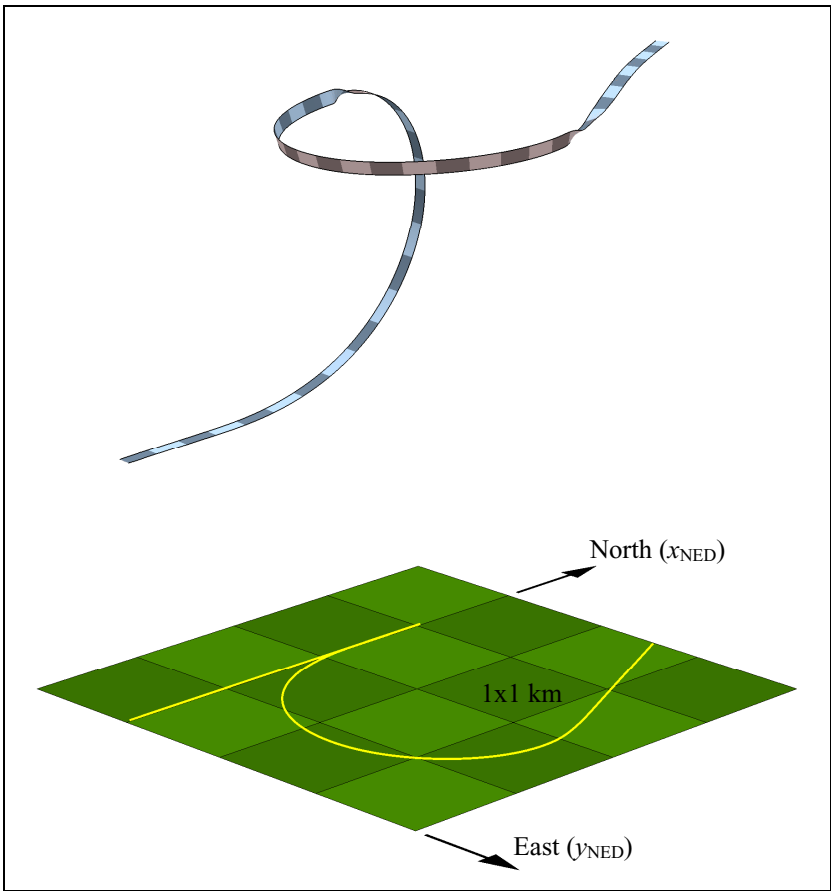


Figure 5.5 Test track number one created using the Hutfly2 simulator. The image is obtained from the results of the inverse simulation.

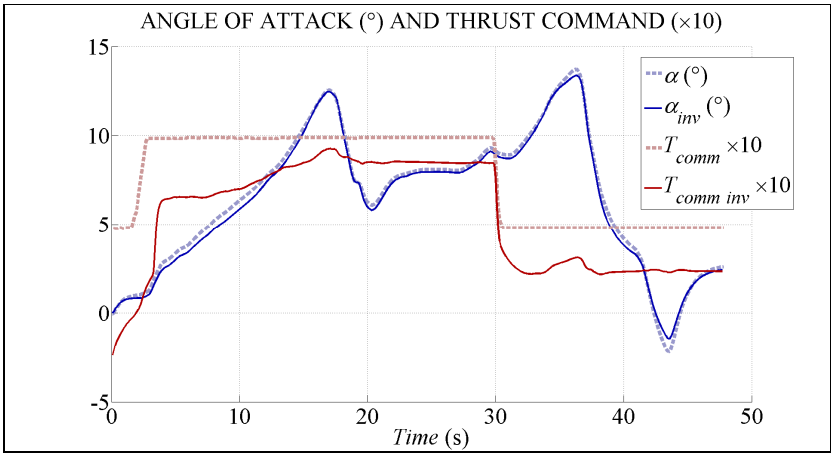


Figure 5.6 Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation.

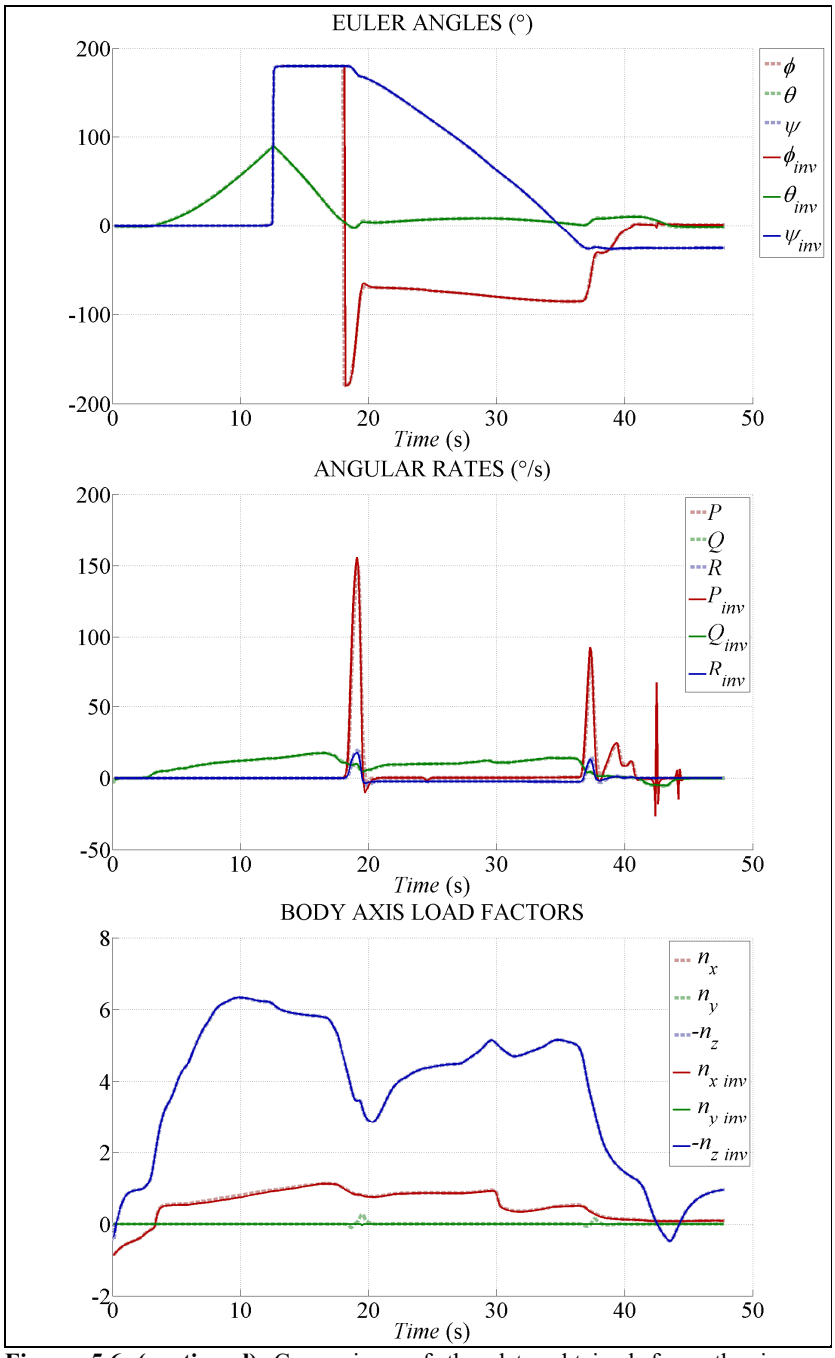


Figure 5.6 (continued) Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation.

The roll time constant of the simplified aerodynamic model was changed from 0.65 to 0.30 to obtain these results, because it was noticed that with the original value the aircraft was unable to stop the rolling motion fast enough at 19 seconds, which led to roll oscillation. Even with the value of 0.30, there is a small overshoot visible in the graph of ϕ_{inv} just before the 20-second mark.

The track includes a momentary negative load factor at around 43 seconds and the negative angle of attack is correctly recognized. Small spikes are visible in the graph of ϕ_{inv}, where the load factor changes to negative and back to positive. The spikes are caused by the presence of a small sideslip angle, and the way they are created was explained in detail in Section 4.2. The spikes are more pronounced in the graph of the angular rates. As the error in the angle of attack is not very big, the Euler angles, angular rates and load factors are reconstructed to a good accuracy. The assumption of coordinated flight seems reasonable when sideslip is not intentionally induced. It can also be seen from the load factor graph (with exceptions at points of high roll rate) that the side load factor n_y of the original data is practically zero. This also indicates that the side force and sideslip are sufficiently small to be neglected.

Actual flight data may be of a considerably lower frequency than the 10 Hz used above. Therefore, the behaviour of the inverse simulation was examined by sampling the first test track also using a lower frequency of 1 Hz. The results of the inverse simulation are shown in Figure 5.7. The inverse simulation works well also with data of lower frequency, but the results are less accurate, especially at the peak values of the angle of attack. This follows from the reduced accuracy of the derivatives which are computed from the differences between successive data points. The curvature of the track is ignored between the points and consequently the distance between successive points is underestimated, especially at strongly curved sections of the track.

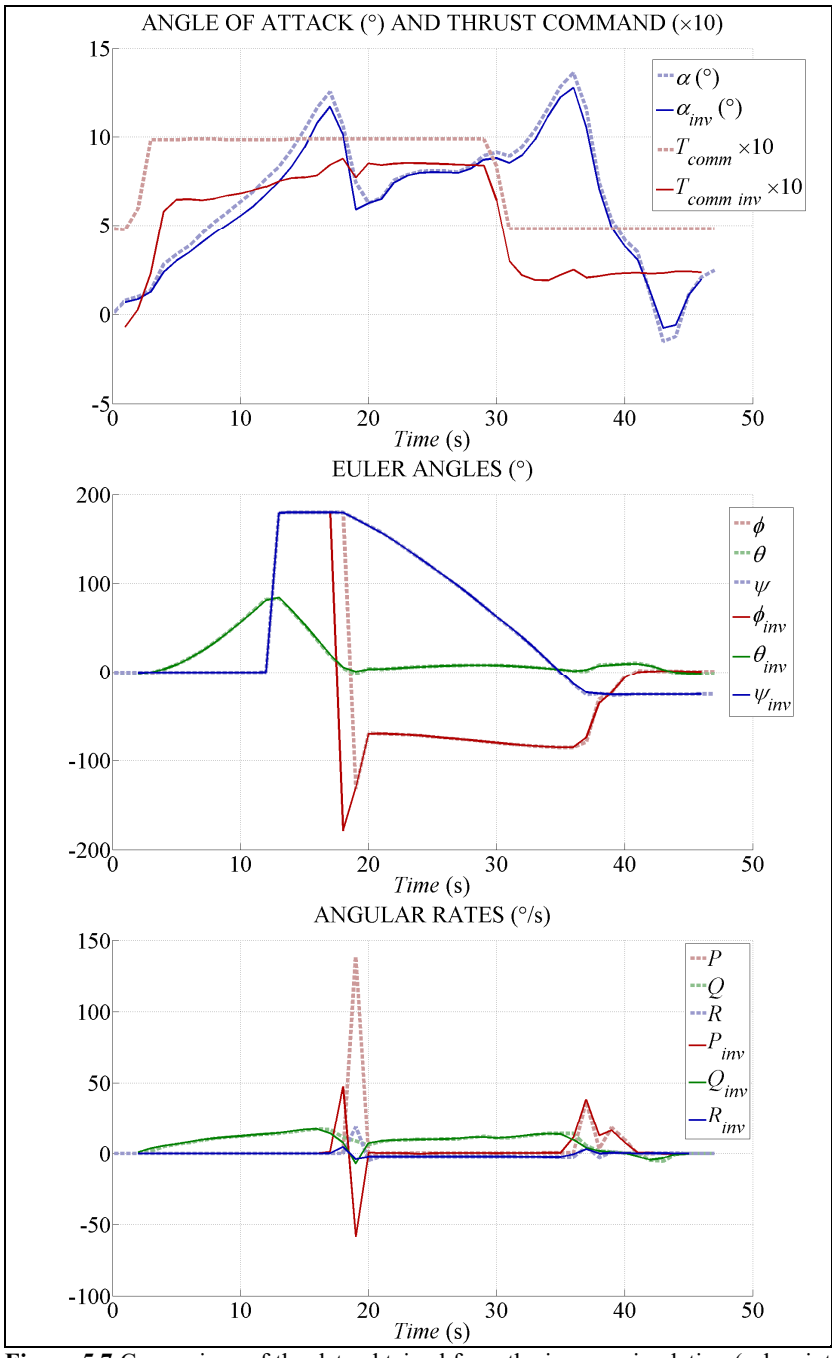


Figure 5.7 Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation (1 Hz input data).

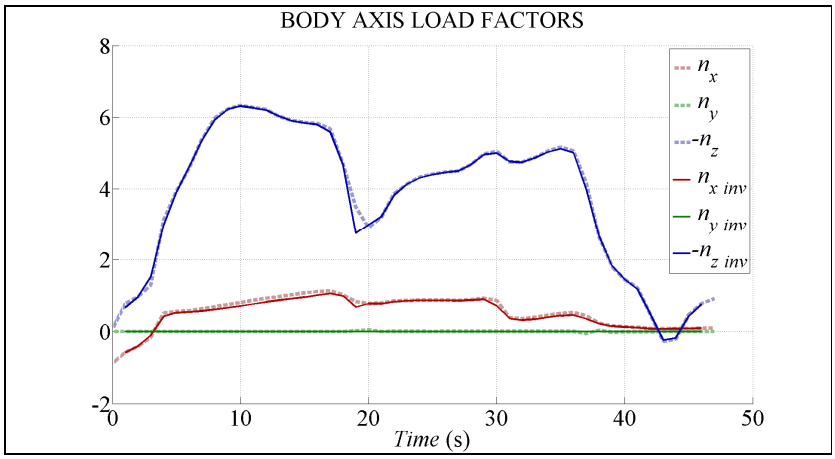


Figure 5.7 (continued) Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation (1 Hz input data).

5.2.2 Test track number two

The second test track (Figure 5.8) is a longer track with a lot of irregular manoeuvring. The results of the inverse simulation are comparable to those of the first track, except within a section in the end where the sign of the load factor changes twice while the sideslip is more visible. The resulting spikes in the graphs are therefore more pronounced, as can be seen in the graphs of Figure 5.9 (only the section in the end of the track is plotted). As described in Section 4.2, the limitation for $\dot{P}$ is disabled for 0.5 seconds right after the sign of the load factor is changed. This can be seen in the graphs of ϕ_{inv} and P as P is allowed to change to its maximum value for a moment to catch up with the orientation of the aircraft. This is to prevent roll oscillation following the induced roll rates right after 61 and 63 seconds, as described in Section 4.2. The presence of the spikes is not a significant problem, because their shape is recognizable, and they can be ignored in the results. The results of the inverse simulation also give a hint of a possible sideslip in the graph of the side load factor $n_{y\ inv}$ right after 63 seconds, but in the absence of side force coefficient data, the corresponding side force is ignored within the inverse simulation, as described in Section 4.2.

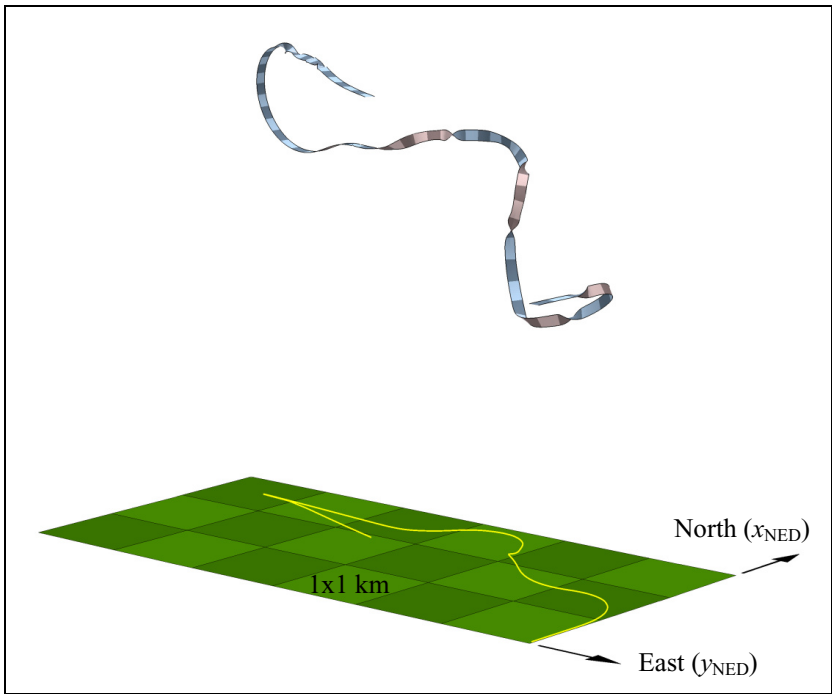


Figure 5.8 Test track number two created using the Hutfly2 simulator. The image is obtained from the results of the inverse simulation.

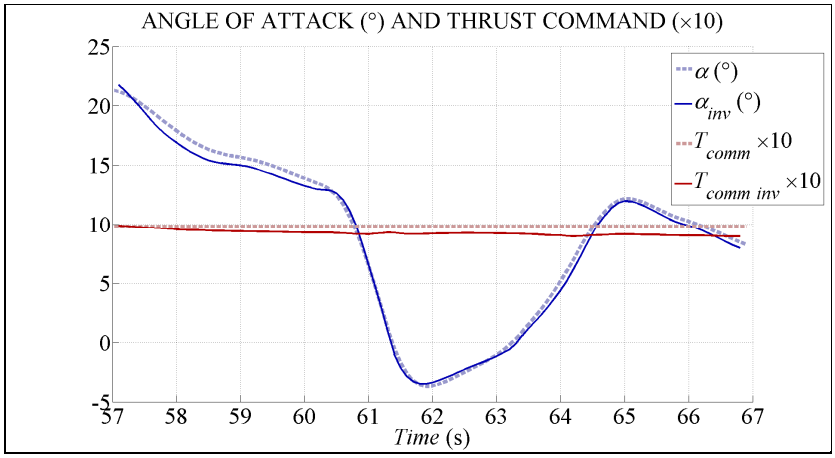


Figure 5.9 Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation.

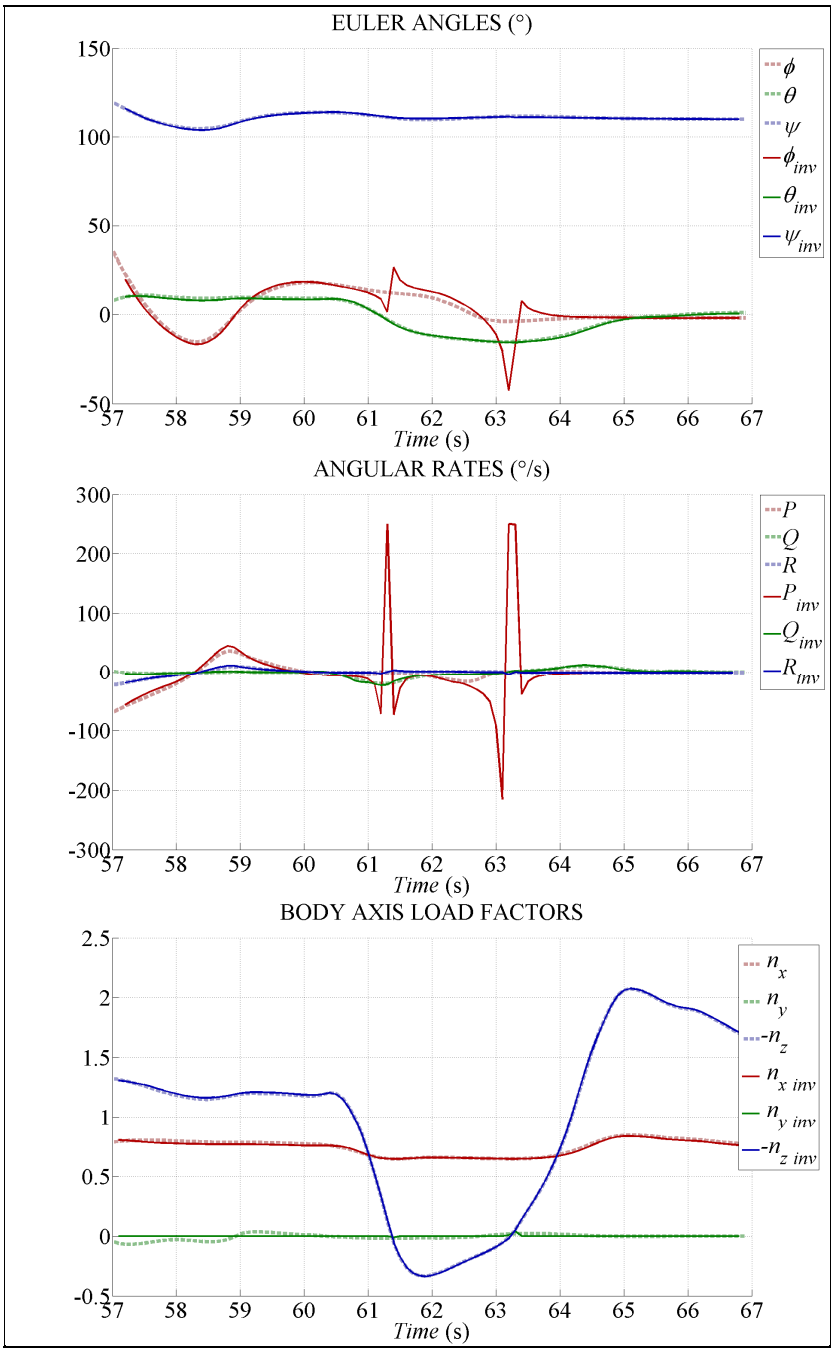


Figure 5.9 (continued) Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation.

5.2.3 Test track number three

The effects of intentional sideslip were studied with the third test track. From straight and level flight, the aircraft does a pull-push-pull manoeuvre, changing the load factor from positive to negative and back to positive (Figure 5.10). Around the latter change of the load factor at 25 seconds, sideslip is intentionally induced using rudder control to turn the nose of the aircraft to the right. As is seen from Figure 5.11, the inverse simulation performs well until the sideslip is induced. This is where the inverse simulation interprets that the aircraft rolls inverted and continues the track in inverted flight. Even though the results are wrong from this point forward, they are also correct in the sense that, as per the simplified aerodynamic model, the aircraft is capable of flying the given track as described by the inverse simulation results.

This is a demonstration of problems that may arise because of multiple solutions: The same track could be flown in many ways. From the different possible solutions, the inverse simulation chooses the one that follows from the assumption of coordinated flight. At 28 seconds, however, there is a visible non-zero side load factor. This is an indication that the aircraft cannot roll fast enough to generate the respective force using the lift of the wing, thus there is a side force component and a respective side load factor left in the results. As described in Section 4.2, the side force component is ignored within the inverse simulation, but it is an indication that at this point of the track the aircraft has not flown in a coordinated way, thus, the results of the inverse simulation are not accurate.

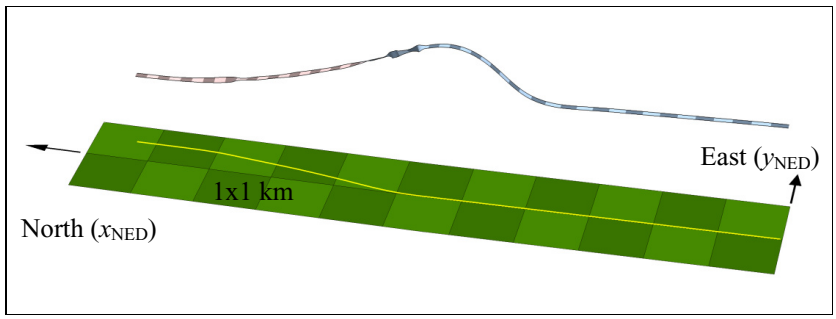


Figure 5.10 Test track number three created using the Hutfly2 simulator. The image is obtained from the results of the inverse simulation.

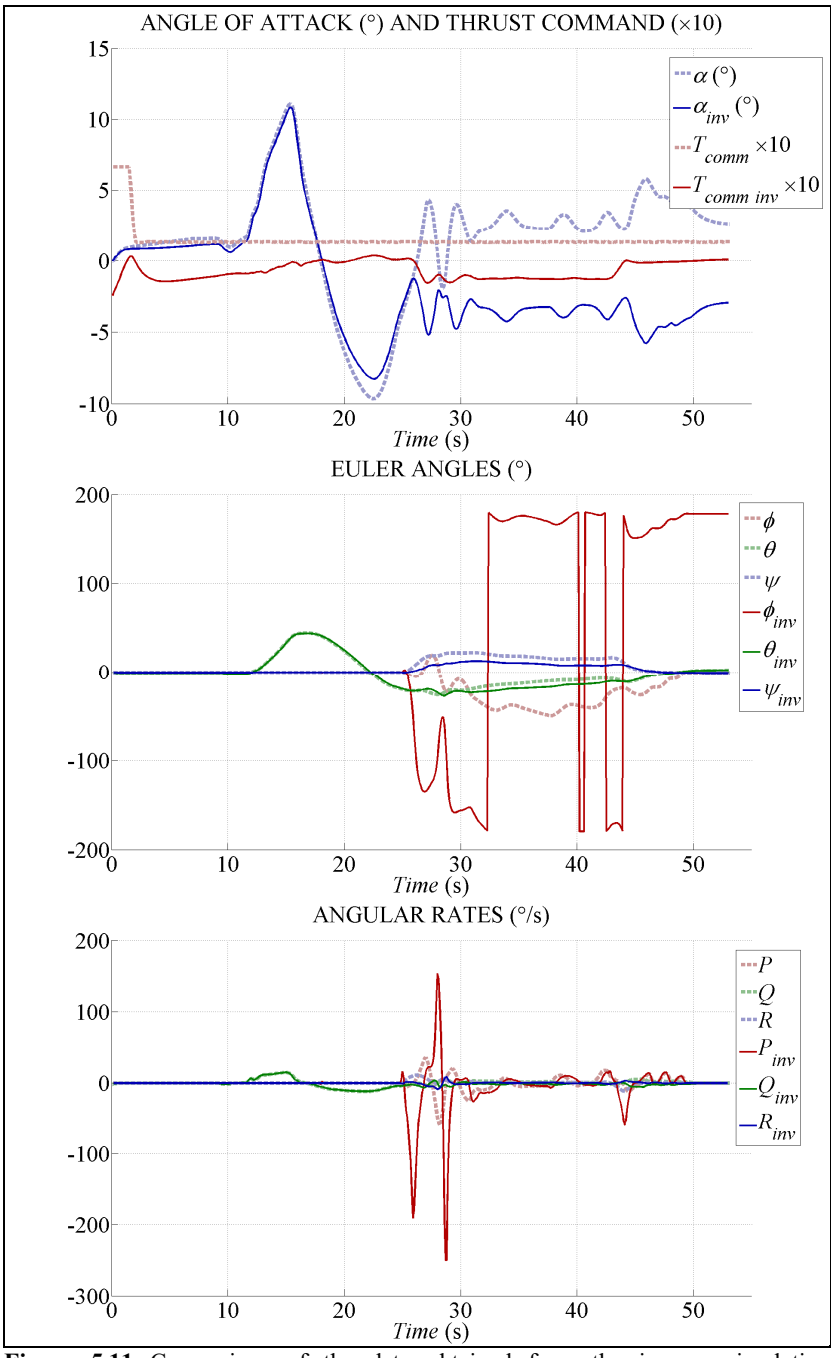


Figure 5.11 Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation.

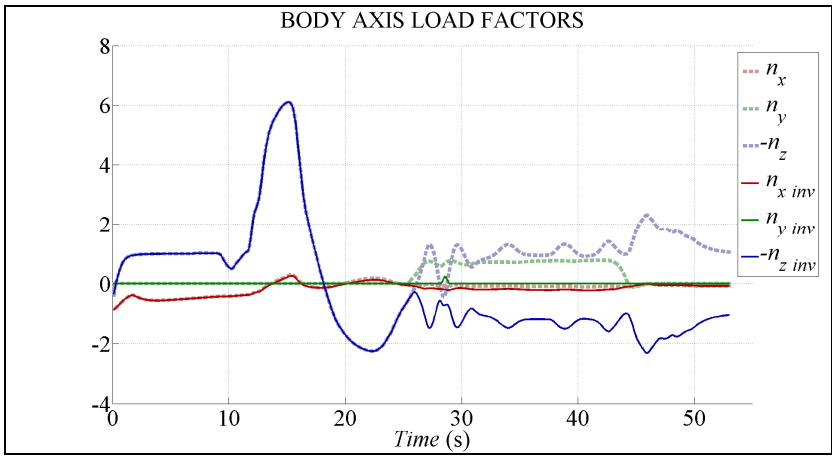


Figure 5.11 (continued) Comparison of the data obtained from the inverse simulation (subscript *inv*) to the original data of Hutfly2 simulation.

5.3 CONCLUSIONS AND DISCUSSION

The inverse simulation performs as expected and can accurately reconstruct the flight parameters, provided that the aircraft has not flown in a sideslip and the initial data are accurate. The initial data are the aerodynamic model, the track data and the atmospheric data, including the wind conditions. If required, further improvements of the algorithm are possible. More intelligence could be added to the algorithm to remove the spikes from the bank angle data at points where the sign of the load factor changes. This could be done by assigning that the bank angle change smoothly from a reliable value before the spike to a reliable value after the spike. In this case, some lateral force affecting the aircraft would be ignored. Alternatively, the lateral force could be used to estimate the sideslip angle if side force coefficient data were available.

Testing of actual flight data is outside the scope of this work, but some considerations can be presented. The frequency of actual flight data may be low and the data may have noise or anomalies. The anomalies such as stray points should be removed, and in the case of noise, adequate smoothing should be carried out to prevent unrealistic values of the derivatives. The inverse simulation itself can be used to test whether the data represents a realistic flight or needs smoothing. As is seen from Subsection 5.2.1, low data frequency reduces the accuracy of the results. To improve the accuracy, the low frequency data points should be connected using smooth curves to obtain a higher frequency representation of the flight track. Finally, if the

wind conditions are not known to a good accuracy, the proportional error in the results of inverse simulation may be twice the proportional error in the wind speed, because lift and drag are proportional to the square of the airspeed.

6 USER INTERFACE AND ANIMATION

6.1 GRAPHICAL USER INTERFACE

Matlab provides a toolbox for creating graphical user interfaces (GUI) and it was utilized to combine the functions of the inverse simulation into an easy-to-use application. Particular attention was paid to the clarity and modularity of the program structure to facilitate easy implementation of possible future modifications and improvements.

The GUI has a number of buttons and input fields (Figure 6.1). The leftmost panel has two options to define the wind conditions: A text file of a format similar to the aerodynamic model can be used to define the wind speed and direction as functions of altitude. Alternatively, a custom Matlab function can be used, which makes it possible to have the wind vary with time for example. Below the wind panel, a button is provided for loading the aerodynamic model of the aircraft from a text file. The mass of the aircraft can be adjusted using the input field provided.

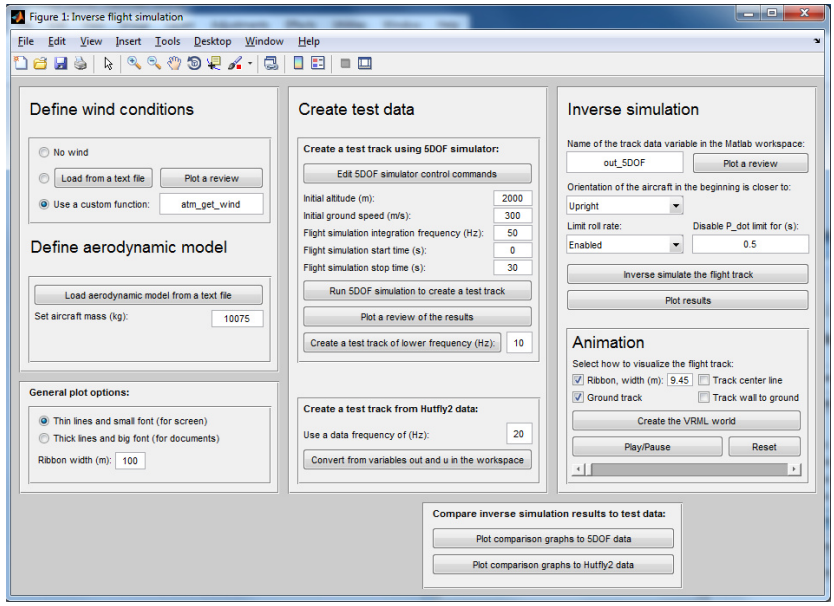


Figure 6.1 The graphical user interface of the inverse flight simulation.

The middle panel can be used to create test data using the 5DOF simulator provided, or by converting data from output variables of Hutfly2. The frequency for sampling the test data can be selected using the respective

input fields. If the sampling frequency is not a simple fraction of the original test data frequency, the closest possible frequency is used.

The rightmost panel controls the actual inverse simulation. A text field is provided for the name of the track data variable. It must be a variable in the Matlab workspace and have the data in four columns (t, x, y, z) with a constant time step. A 3D preview of the track can be plotted via the button next to the text field.

The approximate orientation of the aircraft at the start of the flight track must be known, and a drop-down list is provided to choose the closest from two options: upright and inverted. The user also has an option to disable the limit for the roll rate and to adjust the time to disable the limit for the rate of change of the roll rate. The inverse simulation can be run from the dedicated button, and a progress bar displays the phase of the computation. After the simulation is completed, the results can be displayed in a collection of plots. If the data being inverse simulated is test data, comparison plots can be displayed using the buttons on the bottom of the window. The graphs of Chapter 5 were created using these functions. Some options for the plots can be adjusted via the panel in the lower left corner.

6.2 ANIMATION

The results of the inverse simulation can be viewed as an animation of the aircraft motion. Simple animations can be created within Matlab, which supports viewing of 3D models of the VRML file format. Three VRML files are required to display the animation: The environment, the flight track and the aircraft. The environment includes a simple visualization of the sky and the ground, which has a grid of 1x1 km squares for a reference. The VRML file of the environment has inline calls to the files of the flight track and the aircraft, which can be changed without modifying the file of the environment. When the file of the environment is opened for viewing, the flight track and the aircraft are also visible.

VRML files of some aircraft are available on the internet. For the purpose of this work, a file of the F-16 was copied from the Hutfly2 simulator. The VRML file describing the flight track is created using a custom Matlab function. As VRML files are text files, implementing a Matlab function to create them is straightforward. A button in the animation panel of the GUI creates the flight track file and opens the VRML world in the viewer

application of Matlab. The check buttons of the animation panel can be used to select which components of the flight track visualization are displayed. Figure 6.2 displays the flight track as a ribbon and Figure 6.3 displays the flight track with an altitude visualization.

The animation is implemented as a Matlab function which updates the position and orientation of the aircraft in the VRML world using handles to these objects. The camera positions are saved in the VRML files, and the viewer application can be used to switch between different camera positions and to rotate the view while the animation is running.

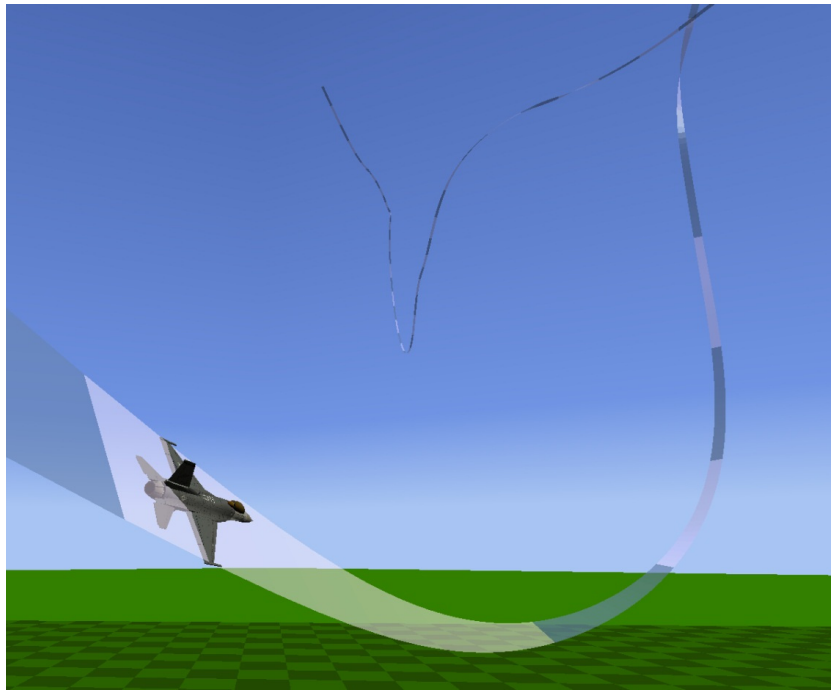


Figure 6.2 Screenshot of the VRML animation: The flight track as a ribbon.

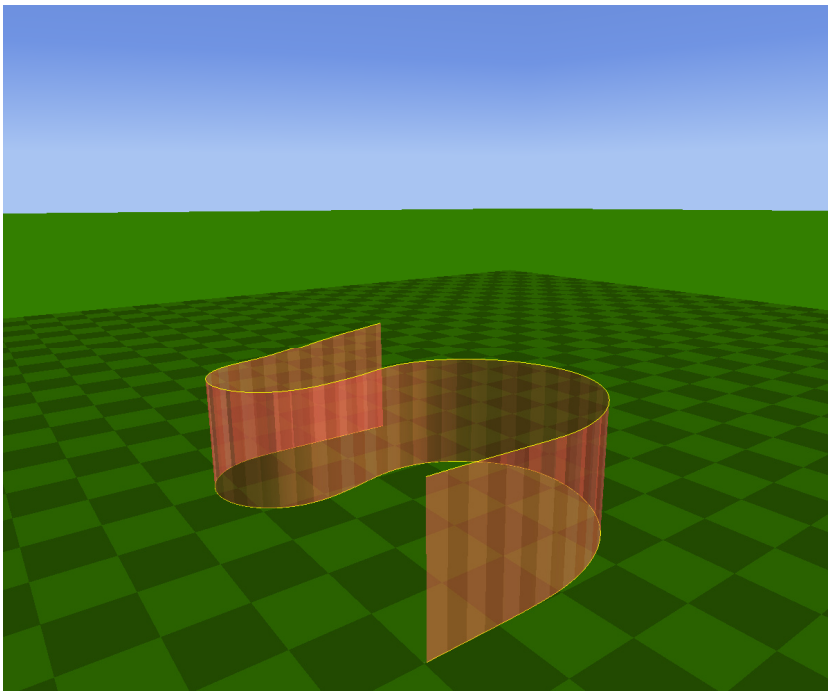


Figure 6.3 Screenshot of the VRML animation: The flight track as an altitude visualization.

7 SUMMARY

In this work, an application for inverse simulation of flight tracks was created. The requirement was to utilize a simplified aerodynamic model to analyse flight tracks that are given in the form of the aircraft position at successive time steps. Such data could be recorded using a GPS device or a radar. The most recognized inverse simulation methods were reviewed, and the method selected for this work is principally similar to the method of Bach and Wingrove [6]. However, the author formulated a different approach, making use of vectors and quaternions, and considered the contribution of the thrust force to the lift. Also, the sections of the flight track flown using a negative load factor are recognized. To obtain a solution, it is assumed that the aircraft flies in a coordinated way, i.e. the sideslip angle and consequently the side force are negligible.

To verify the algorithm of the inverse simulation, a 5DOF flight simulator was implemented. The flight simulator was used to create a test track that perfectly satisfies the assumption of coordinated flight. The test track was used as an input to the inverse simulation, and it was shown that the inverse simulation can accurately reconstruct the flight parameters, i.e. thrust, angle of attack, Euler angles, angular rates and load factors.

The inverse simulation was then tested using more realistic flight tracks, in which the assumption of coordinated flight might not be exactly satisfied. These flight tracks were generated by manually flying a high fidelity flight simulator, Hutfly2. It was shown that results from the inverse simulation are good if the pilot does not intentionally induce sideslip.

The application was implemented in Matlab, including a graphical user interface and a possibility to view an animation of the flight. Particular attention was paid to the clarity and modularity of the program structure, thereby making possible future modifications and improvements straightforward. The application can be used to analyse flight accidents if a record of the aircraft flight path is available. The application can also be used to verify the feasibility of flight tracks that are generated using optimization methods and to support monitoring of aircraft fatigue life through the flight parameters. Actual flight data may have to be processed to remove anomalies and noise to obtain good results from the inverse simulation.

Possible future improvements could include estimation of sideslip angle at points of the flight track where the sideslip has to be considered to obtain an accurate solution. This would require side force coefficient data of the aircraft. The availability of a more detailed aerodynamic model of the aircraft would enable further development to add estimation of control surface deflections.

REFERENCES

[1] Öström, J., Inverse Flight Simulation for a Fatigue Life Management System, AIAA 2005-6212, AIAA Modeling and Simulation Technologies Conference, San Francisco, CA, 2005.

[2] Sailaranta, T., The use of GPS positioning data – Final Report, Report T-263, Aalto University, Aerodynamics Research Unit, Espoo 2008. (in Finnish, classified)

[3] Öström, J., Hoffren J., Inverse Flight Simulation Using Aircraft Performance Models, AIAA 2006-6822, AIAA Modeling and Simulation Technologies Conference, Keystone, CO, 2006.

[4] Karelahti J., Virtanen K., Öström J., Automated Generation of Realistic Near-Optimal Aircraft Trajectories, AIAA Journal of Guidance, Control, and Dynamics, Vol. 31, No. 3, 2008.

[5] Sailaranta, T., Method for Inverse Analysis of Flight of Missile, Report T-258, Aalto University, Aerodynamics Research Unit, Espoo 2008. (in Finnish, not published)

[6] Bach R., Wingrove R., Equations for Determining Aircraft Motions from Accident Data, NASA-TM-78609, Ames Research Center, Moffett Field, CA, June 1980.

[7] Öström, J., Hutfly2, Matlab / Simulink based Flight Simulation Software, Report B-56, Aalto University, Aerodynamics Research Unit, Espoo 2005. (in Finnish) Availability as of December 2011: http://www.aero.hut.fi/pubs/reports/B-56.pdf

[8] Thomson D., Bradley R., Inverse simulation as a tool for flight dynamics research — Principles and applications, Progress in Aerospace Sciences 42, 2006.

[9] Kato O., Sugiura I., An interpretation of airplane motion and control as inverse problem, AIAA Journal of Guidance, Control, and Dynamics, Vol. 9, No. 2, 1986.

[10] Thomson D., An analytical method of quantifying helicopter agility, Paper 45, Proceedings of the 12th European Rotorcraft Forum, Garmisch-Partenkirchen, Federal Republic of Germany, 1986.

[11] Hess, R.A., Gao, C., Wang, S.H., Generalized Technique for Inverse Simulation Applied to Aircraft Maneuvers, AIAA Journal of Guidance, Control, and Dynamics, Vol. 14, No. 5, 1991.

[12] Matteis G., de Socio L.M., Leonessa A., Solution of Aircraft Inverse Problems by Local Optimization, AIAA Journal of Guidance, Control, and Dynamics, Vol. 18, No. 3, 1995.

[13] Avanzini G., de Matteis G., Two-timescale-integration method for inverse simulation, AIAA Journal of Guidance, Control, and Dynamics, Vol. 22, No. 3, 1999.

[14] Stevens B.L., Lewis F.L., Aircraft Control and Simulation, John Wiley & Sons Inc., 1992, ISBN 0-471-61397-5

[15] Phillips W.F., Hailey C.E., Gebert G.A., Review of Attitude Representations Used for Aircraft Kinematics, Journal of Aircraft, Vol. 38 No. 4, 2001.

[16] Vilenius J., Structure of Standardized Aircraft Model, Aalto University, Aerodynamics Research Unit, Espoo 1996. (in Finnish, not published)

[17] Properties of a standard atmosphere, ESDU 77021, Engineering Sciences Data Unit, London, 1986

[18] Standard Atmosphere – Tables and Data, NACA Report 1235, 1955

[19] Maths - Rotation conversions, website www.euclideanspace.com, Availability as of December 2011: http://www.euclideanspace.com/ maths/ geometry/rotations/conversions/index.htm

APPENDIX A

MATRIX AND QUATERNION ALGEBRA [15,19]

Multiplication of two quaternions is accomplished as follows:

$$\mathbf{ab} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} a_1 b_1 - a_2 b_2 - a_3 b_3 - a_4 b_4 \\ a_1 b_2 + a_2 b_1 + a_3 b_4 - a_4 b_3 \\ a_1 b_3 - a_2 b_4 + a_3 b_1 + a_4 b_2 \\ a_1 b_4 + a_2 b_3 - a_3 b_2 + a_4 b_1 \end{bmatrix} \tag{A.1}$$

Conjugate of a quaternion is

$$\mathbf{q}^* = \begin{bmatrix} q_1 \\ -q_2 \\ -q_3 \\ -q_4 \end{bmatrix} \tag{A.2}$$

Reciprocal (inverse) of a quaternion is

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \tag{A.3}$$

If **q** is a unit quaternion, it follows that

$$\mathbf{q}^{-1} = \mathbf{q}^* \tag{A.4}$$

A rotation of an $\mathbb{R}^3$ vector **v** using a quaternion **q** is accomplished as follows, by treating **v** as a quaternion with a first component equal to zero:

$$\mathbf{v}' = \mathbf{q}\mathbf{v}\mathbf{q}^{-1} = \mathbf{q} \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} \mathbf{q}^{-1} = \begin{bmatrix} 0 \\ v'_1 \\ v'_2 \\ v'_3 \end{bmatrix} \tag{A.5}$$

A quaternion representing a rotation can be obtained from the rotation angle φ and the unit vector **u** of the rotation axis as follows:

$$\mathbf{q} = \begin{bmatrix} \cos(\varphi/2) \\ u_1 \sin(\varphi/2) \\ u_2 \sin(\varphi/2) \\ u_3 \sin(\varphi/2) \end{bmatrix} \tag{A.6}$$

To obtain a quaternion from a rotation matrix, a method described in Ref. [19] is used. The basic equation is

$$\mathbf{q} = \begin{bmatrix} s/4 \\ (R_{32} - R_{23})/s \\ (R_{13} - R_{31})/s \\ (R_{21} - R_{12})/s \end{bmatrix} \tag{A.7}$$

where

$$s = 2\sqrt{1 + R_{11} + R_{22} + R_{33}} \tag{A.8}$$

Rather than the equation, a function that deals with possible singularities and numerical issues is used. The issues are discussed in Ref. [19] and the function is implemented in Matlab as follows:

```
function [ q ] = q_from_rotation_matrix( R )
%Q_FROM ROTATION_MATRIX This requires a pure rotation matrix 'R' as input.
%
%   Code is copied from www.euclideanspace.com
%   and converted to Matlab syntax.

    q = zeros(4,1);
    tr = R(1,1) + R(2,2) + R(3,3);

    if tr > 0
        S = sqrt(tr+1.0) * 2; %S=4*qw
        q(1) = 0.25 * S;
        q(2) = (R(3,2) - R(2,3)) / S;
        q(3) = (R(1,3) - R(3,1)) / S;
        q(4) = (R(2,1) - R(1,2)) / S;

    else if R(1,1) > R(2,2) && R(1,1) > R(3,3)
            S = sqrt(1.0 + R(1,1) - R(2,2) - R(3,3)) * 2; %S=4*qx
            q(1) = (R(3,2) - R(2,3)) / S;
            q(2) = 0.25 * S;
            q(3) = (R(1,2) + R(2,1)) / S;
            q(4) = (R(1,3) + R(3,1)) / S;

        else if (R(2,2) > R(3,3))
                S = sqrt(1.0 + R(2,2) - R(1,1) - R(3,3)) * 2; %S=4*qy
                q(1) = (R(1,3) - R(3,1)) / S;
                q(2) = (R(1,2) + R(2,1)) / S;
                q(3) = 0.25 * S;
                q(4) = (R(2,3) + R(3,2)) / S;
```

```
            else
                S = sqrt(1.0 + R(3,3) - R(1,1) - R(2,2)) * 2; %S=4*qz
                q(1) = (R(2,1) - R(1,2)) / S;
                q(2) = (R(1,3) + R(3,1)) / S;
                q(3) = (R(2,3) + R(3,2)) / S;
                q(4) = 0.25 * S;

            end
        end
    end
end
```

(A.9)

To obtain the rotation angle and unit axis from a rotation matrix, a method of Ref. [19] is also used. The basic equations are

$$\varphi = \arccos\left(\frac{R_{11}+R_{22}+R_{33}-1}{2}\right) \tag{A.10}$$

$$\mathbf{u} = \begin{bmatrix} (R_{32}-R_{23})/s \\ (R_{13}-R_{31})/s \\ (R_{21}-R_{12})/s \end{bmatrix} \tag{A.11}$$

where R_{ij} is an element of the rotation matrix on the line i and column j and

$$s = \sqrt{(R_{32}-R_{23})^2+(R_{13}-R_{31})^2+(R_{21}-R_{12})^2} \tag{A.12}$$

The Matlab code of a function which deals with possible singularities and numerical issues is given here:

```
function [ axis, angle ] = axis_angle_from_matrix( R )
%AXIS_ANGLE_FROM_MATRIX This requires a pure rotation matrix 'R' as input.
%
%   Code is copied from www.euclideanspace.com
%   and converted to Matlab syntax.

    epsilon = 0.000001; % Margin to allow for rounding errors
    epsilon2 = 0.00001; % Margin to distinguish between 0 and 180 degrees


    if     abs( R(1,2)-R(2,1) ) < epsilon ...
        && abs( R(1,3)-R(3,1) ) < epsilon ...
        && abs( R(2,3)-R(3,2) ) < epsilon

        % Singularity found. First check for identity matrix which must
        % have +1 for all terms in leading diagonal and zero in other terms
        if     abs(R(1,2)+R(2,1)) < epsilon2 ...
            && abs(R(1,3)+R(3,1)) < epsilon2 ...
            && abs(R(2,3)+R(3,2)) < epsilon2 ...
            && abs(R(1,1)+R(2,2)+R(3,3)-3) < epsilon2

            % This singularity is identity matrix so angle = 0
            angle = 0;
            axis  = [1 0 0]';   % Return arbitrary axis
```

```
    else

        % Otherwise this singularity is angle = 180
        angle = pi;
        xx = (R(1,1)+1)/2;
        yy = (R(2,2)+1)/2;
        zz = (R(3,3)+1)/2;
        xy = (R(1,2)+R(2,1))/4;
        xz = (R(1,3)+R(3,1))/4;
        yz = (R(2,3)+R(3,2))/4;

        if xx > yy && xx > zz   % R(1,1) is the largest diagonal term

            if xx < epsilon
                x = 0;
                y = 0.70710678;
                z = 0.70710678;
            else
                x = sqrt(xx);
                y = xy/x;
                z = xz/x;
            end

        else

            if yy > zz          % R(2,2) is the largest diagonal term

                if yy < epsilon
                    x = 0.70710678;
                    y = 0;
                    z = 0.70710678;
                else
                    y = sqrt(yy);
                    x = xy/y;
                    z = yz/y;
                end



            else                % R(3,3) is the largest diagonal term

                if zz < epsilon
                    x = 0.70710678;
                    y = 0.70710678;
                    z = 0;
                else
                    z = sqrt(zz);
                    x = xz/z;
                    y = yz/z;
                end

            end

        end

        axis  = [x y z]';   % Return arbitrary axis

    end

else

    % As we have reached here there are no singularities so we can
    % handle normally. s is used to normalize.
    s = sqrt(    (R(3,2) - R(2,3)) * (R(3,2) - R(2,3)) ...
              + (R(1,3) - R(3,1)) * (R(1,3) - R(3,1)) ...
              + (R(2,1) - R(1,2)) * (R(2,1) - R(1,2))     );

    % Prevent divide by zero, should not happen if matrix is orthogonal
    % and should be caught by singularity test above, but left it in
    % just in case:
    if abs(s) < 0.001
        s=1;
    end

    % Compute angle and axis using the basic equation
    angle = acos( (R(1,1) + R(2,2) + R(3,3) - 1) / 2 );
```

```
            x = (R(3,2) - R(2,3)) / s;
            y = (R(1,3) - R(3,1)) / s;
            z = (R(2,1) - R(1,2)) / s;
            axis  = [x y z]';

        end

    end
```

(A.13)

In this work, a method and an application are created for inverse simulation of flight track data that are given in the form of the aircraft position over time. Such data could be recorded using a GPS-device or a radar. A number of essential flight parameters can be reconstructed via the inverse simulation, e.g. Euler angles, angular rates, angle of attack and thrust force. The inverse simulation requires a simplified aerodynamic model of the aircraft and information about the wind conditions. The results can be viewed as a set of plots, or as an animation of the aircraft motion. The application can be used to analyse flight accidents and to assess the feasibility of flight tracks that are generated using optimization methods. The application could also be used to support monitoring of aircraft fatigue life through the flight parameters. The work includes an implementation of a five-degree-of-freedom flight simulator, which is used to verify the algorithm of the inverse simulation.

ISBN 978-952-60-4541-2
ISBN 978-952-60-4542-9 (pdf)
ISSN-L 1799-4896
ISSN 1799-4896
ISSN 1799-490X (pdf)

Aalto University
School of Engineering
Department of Applied Mechanics
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS

Aalto-ST 8/2012