Reconfigurable Control Allocation Design with Applications to Unmanned Aerial Vehicle and Aircraft

Qing-Li Zhou

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirement

for the Degree of Master of Applied Science (Mechanical Engineering) at

Concordia University

Montreal, Quebec, Canada

August 2009

Your file Votre référence
ISBN: 978-0-494-63090-7
Our file Notre référence
ISBN: 978-0-494-63090-7

ABSTRACT

Reconfigurable Control Allocation Design with Applications to Unmanned Aerial Vehicle and Aircraft

Qing-Li Zhou

The main objective of this thesis is to design and evaluate reconfigurable flight control system against control surfaces faults in Unmanned Aerial Vehicle (UAV) and aircraft without modifying the baseline controller/control law by using control re-allocation technique. The faults are introduced in the form of partial loss and stuck at unknown positions of control surfaces on the UAV and aircraft. Four control reallocation algorithms with applications to UAV and fixed-wing aircraft were investigated, which include a pseudo-inverse, a fixed-point algorithm, a direct control allocation algorithm and a weighted least squares method. The thesis work is evaluated by a nonlinear UAV model ALTAV (Almost-Light-Than-Air-Vehicles), developed by Quanser Inc., and a nonlinear aircraft model ADMIRE (Aero-Data-Model-in-Research-Environment), developed by the Group of Aeronautical Research and Technology in Europe (GARTEUR). Different faults have been introduced in control surfaces with different commanded inputs. Gaussian noise was introduced in the ALTAV model. Different faults have been introduced in control surfaces with different command inputs. Comparisons were made under normal situation, fault conditions without control re-allocation, and with control reallocation. Simulation results show the satisfactory reconfigurable flight control system performance using control re-allocation methods for ALTAV UAV model and ADMIRE aircraft model.

Acknowledgement

Many people have contributed in a variety of ways to the production of this thesis. First of all, I am grateful to my supervisor, Professor Youmin Zhang, for giving me the opportunity to perform this research at Concordia University and for his continuous guidance and support. His inspiring advice and encouragement has guided me during this thesis. This work would have never been done without his great vision, experience and insight. I also highly appreciate his giving me great freedom in selecting my research topics.

Also, I would like to thank all my friends and colleagues here, at Concordia University, who have been a source of great strength to me.

I want to express my gratitude to the Quanser incorporation and GARTEUR group for giving me the simulation software and information on the baseline. And, special thanks to Concordia University for providing the facilities to complete this work.

Last, but not least, I am sincerely thankful to my parents and my husband, Xiang-Dong, for their encouragement and incredible support. Especially, I thank Xiang-Dong for being so generous in taking care of our kids and letting me put more time on finishing my studies. Here, I also thank my two kids for their loveliness and obedience. I am lucky to have such a wonderful, lovely and supportive family.

This thesis is dedicated to my parents and my husband,

Xiang-Dong,

For their love and understanding.

Table of Contents

List of Figures

1. Introduction

An Unmanned Aerial Vehicle (UAV) is an aircraft that is driven by power. It can fly without an on-board operator and can be re-used and is re-usable. One of the most appealing topics among the research control community is the application of modern control theory to UAVs. Such vehicles can be controlled remotely by an operator on the ground, or autonomously via a pre-designed program. Interest in using UAVs is due to their wide-range field applications, both civil and military. Applications like traffic surveillance, area mapping and forest fire detection require high manoeuvrability of the aircraft and the robustness of the control algorithm with respect to parameter uncertainties and disturbances like wind and weather condition changes. UAV is playing an increasingly important role in modern high technology thanks to its unique characteristics [1]. However, due to the lack of a pilot, UAV loses the human ability of making smart decisions. This may lead to mission failure when UAV operates in abnormal conditions, such as flight computer failure, airborne sensor failure and control surface damage. On the other hand, UAV in war needs to have a good performance to escape the opponent's attack. From domestic and foreign high reliability flight control system development, the UAV flight control system requires high reliability and high survivability to maximally ensure the safety of the UAV and equipment, in order for the UAV to safely complete reconnaissance and surveillance missions. The same as UAV, all modern airplanes with pilots depend upon their flight control systems to provide the handling qualities necessary for successful flight. Therefore, it is necessary to develop flight control systems that can enable aircrafts to successfully complete missions in the

presence of non-fatal fault cases through reconfigurable flight control system design. Some previous research works were introduced in [5, 6, 10].

As one part of this thesis, the Quanser Almost-Lighter-Than-Air Vehicle (ALTAV) UAV model is used in simulation studies. The ALTAV uses buoyancy to float in the air in a way that is similar to ships floating in water. The Quanser ALTAV [2, 4] provides a platform to demonstrate control reallocation methods for unmanned aerial vehicles.

As the other part of this thesis, the GARTEUR Aero-Data Model [19] in Research Environment (ADMIRE) aircraft model is also evaluated. This benchmark model provides a realistic, nonlinear, fixed-wing aircraft model.

For using these two benchmarks, partial loss and stuck faults have been implemented for reconfigurable flight control system design. Four reconfigurable control allocation methods, pseudo-inverse, fixed-point, direct control allocation and weighted least squares, are used and evaluated. These techniques meet the challenges for the partial loss and stuck fault on control surfaces in simulations.

1.1. Fault Tolerant Control System

With the developments in control systems, high reliability, availability and safety have become important requirements, included in many international standards and regulations. Besides the quality and robustness, use of hardware redundancy is a traditional way to improve process reliability and availability, which has been extended to the use of software redundancy during the last decades.

The Fault-Tolerant Control System (FTCS), also known as fail-safe control system, is a

control system that possesses the ability to accommodate for system failures automatically and to maintain overall system stability and acceptable performance in the event of component failures [23]. The objective of FTCS is to maintain safety and reliability of modern engineering systems. Typically, a FTCS consists of three parts: a reconfigurable controller, which includes baseline controller and control allocation modules, a Fault Detection and Identification (FDI) scheme, and a control law reconfigurable mechanism. This thesis will focus on the development of the reconfigurable control allocation technique.

The importance of reconfigurable control allocation has now attracted wide attention, especially the signal treatment, pattern recognition, adaptive control, optimization and intelligent control; the integration of these techniques under the concept of reconfigurable control has sped up the reconstruction of the flight control system technology.

Generally, relying on information from the fault detection and diagnosis, reconfigurable control allocation can be classified into two categories:

(1) Reconstruction methods relying on FDI information (Active)

These methods refer to the use of a variety of information of prior failure and impact; a pre-established program of reconstruction stored in the onboard computer, when fault happens, redistribution control system commands according to the results of FDI to offset the impact of control surface stuck or compensate the impact of partial loss; effective use of the remaining control surfaces to complete missions or to ensure safe landing; the stability and acceptable performance of the entire system can be maintained. In certain circumstances, degraded performance may have to be accepted. Also, this method can calculate control reconfiguration on-line and re-distribute the

control commands [3, 7, 8, 12, 13, 15].

(2) Control not relying on FDI information (Passive)

In contrast to (1), these methods are also known as reliable control systems. System components and controllers are designed to be robust against a class of presumed faults. At the beginning of the design phase, the main goal in a fault-tolerant control system is to design components and controllers with a suitable structure to achieve stability and satisfactory performance, not only when all control components are functioning normally, but also in cases when there are malfunctions in sensors, actuators, or other system components.

This robust flight control system can maintain stability in the presence of faults. This approach needs neither FDI scheme nor controller reconfiguration, but it has limited fault-tolerant capabilities [16-18]. Discussions on passive FTCS are beyond the scope of this thesis.

Following the wide-range development of UAV, the demand for higher system performance, quality and cost efficiency leads to the growth of complexity in control systems. In control systems, severe faults happen such as actuator or sensor outages; producing a break-up of the control loop, which must be restructured to prevent failure at the system level. Control allocation reconfiguration is an active approach in control theory to achieve fault-tolerant control of dynamic systems, without need to restructure the controller parameter. Control reconfiguration is a building block toward increasing flight dependability.

The structure diagram of a fault-tolerant control system is shown below (not considering the FDI block).

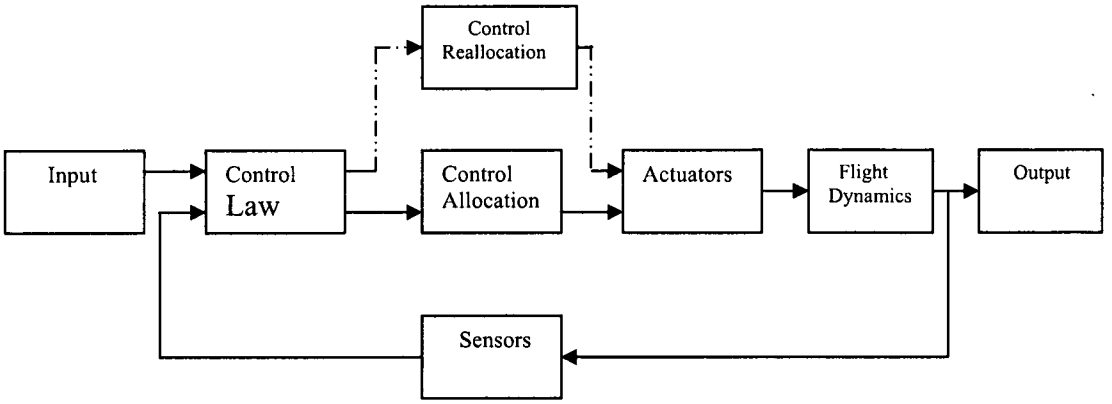

Figure 1. 1 Simple structure of fault tolerant control system

Figure 1.1 shows that the control allocation block and control reallocation block are placed between the control law and actuators. The system uses the control allocation block when the system operates in normal conditions; the system uses the control reallocation block after it has faults caused by control surfaces or actuators. The algorithms implemented into the control allocation and reallocation blocks must be chosen amongst many different constrained optimization based algorithms. These include, but are not limited to: least-squares, linear programming and quadratic programming. The simplest control allocation method is based on the unconstrained least squares algorithm with small modifications to consider position limits of the actuators. More complex methods are derived from the constrained least squares optimization to solve the control allocation problem. Until recently, it was believed that control allocation was too complex and computationally intensive for real world use in flight control cases. However, the recent, dramatic improvement in computer speed and the development of more efficient algorithms have changed the situation considerably.

This led to the development of different fault tolerant control techniques. In this thesis,

fault tolerant control of ALTAV is achieved based on the following control reallocation techniques:

1. Pseudo-inverse (Pinv)
2. Fixed-point (Fix)
3. Direct Control Allocation (DCA)
4. Weighted Least Square(WLS)

1.2. The Main Content of The Thesis

Reconstruction technology research of control system is based on the redundancy configuration and control redundancy ability to manipulate control surfaces, guarantee UAV and aircraft survival ability at a non-serious fault. This project mainly studies UAV and aircraft control redundancy ability to control manipulating surfaces, the ability to track the desired trajectory path mission and safe return despite a minor fault and the ability to work well and safe return when a serious failure happens. The reconfigurable control allocation methods can ensure UAV and aircraft stability under incomplete or not completely manipulated information. As to redundant control manipulation surfaces, when one of the manipulation surfaces has a malfunction or damage, UAV and aircraft can increase the residual amount of remaining control surfaces to achieve stable flight control effects.

The structure of this thesis is as follows. The second chapter presents the basic concept, theory of control allocation and two fault models. The third chapter presents four different reconfigurable control methods, not only explaining these methods in theory, but

also gives example to explain. Then, the fourth chapter describes the basic dynamics of the ALTAV UAV and benchmark model. The fifth chapter describes the ADMIRE aircraft model. The sixth chapter presents the implementation of control allocation methods in ALTAV UAV and ADMIRE aircraft benchmarks. Then, the seventh chapter presents ALTAV simulation results of different types and in different circumstances, while the eighth chapter presents ADMIRE results. The ninth chapter is a conclusion of my work, summarize my contribution and possible future work.

2. Flight Control Allocation

Control allocation is useful for control of over-actuated systems; control allocation is concerned with how to distribute the deflection of multiple control surfaces of the aircraft to generate the required control inputs, including heading, pitch, roll moments, and forces, when the number of control surfaces is greater than the number of required control inputs. Using control allocation, the actuator selection task is separated from the regulation task in the control design. The control allocation problem is studied following the work of Durham [21].

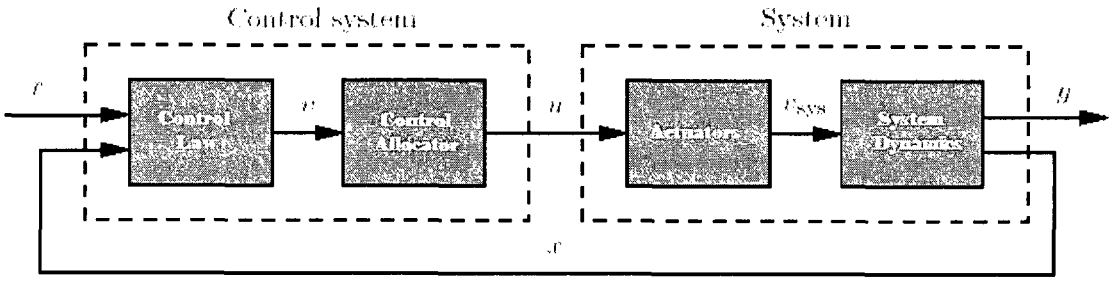

Figure 2. 1 Control allocation block diagram

Figure 2.1 is the control system structure when control allocation is used. The control system is made up of a control law, specifying the total control effect, v should be produced and a control allocator, which distributes this control demand among the individual actuators, u is produced. In this system, actuators generate a total control effect v_{sys}, which determines the system behavior. If the control allocation is successful, $v_{sys} = v$.

Today, control allocation is a research topic in aerospace control, marine vessel control and UAV control.

It can also be seen from Figure 2.1 that the flight control system can be split into two parts: (1) control law and control allocation. That is to say that there are two control systems which should be re-designed to maintain the stability and desired transient and steady-state performance when the actuator, sensor or computer have a fault. In this thesis, reconfigurable control law is out of our scope. The main focus will be placed on reconfigurable control allocation or termed alternatively as control re-allocation. This control re-allocation block is placed between the control law and the actuators. There are some benefits that lead us to separate the control re-allocation block:

1) Actuator constraints can be taken into account. If one actuator saturates and fails to produce similar or close to the control effect under nominal conditions, the remaining functional actuators may be used to make up the difference.
2) Reconfiguration can be performed if the effectiveness of the actuators changes over time, or in the event of an actuator failure, without having to redesign the control law.
3) Actuator utilization can be treated independently and can be optimized for the application considered.

The purpose of this work is to provide different algorithms to re-allocate the control effectors (control surfaces or actuators) in overall FCS by redistributing the control effect to the remaining healthy control surfaces, and at the same time, canceling the effects of control surface deflection stuck at neutral or non-neutral position, as well as partial loss, by generating a compensated control signal to be finally fed into the FCS.

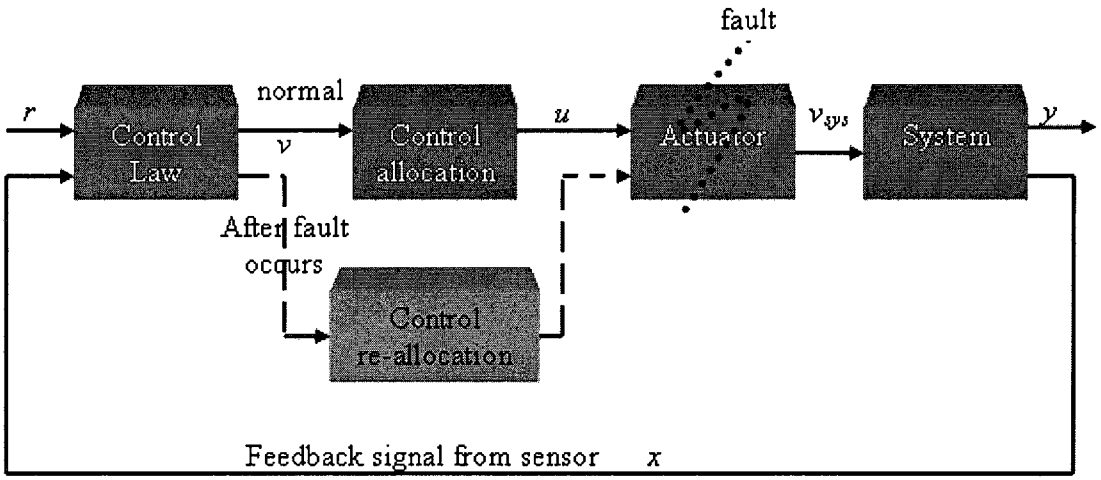

Figure 2. 2 Control re-allocation block diagram

The control re-allocation algorithms use the redundancy of the control allocation technique until the actuators get saturated. These algorithms are the same as used in the control allocation block. The idea is, for a normal situation (without fault), the signals from the control law go through the control allocation block and pass to the actuator, then to the system. When fault occurs, the signals from the control law pass through the control re-allocation block and to the actuator. The algorithms applied in the control re-allocation block re-distribute the control signals calculated by the baseline control law in the presence of faults and send the redistributed signals to each control surface to achieve acceptable performance.

2.1. Introduction to Control Allocation

Control allocation is frequently described depending on the application. In this section, effort is made to develop a generic mathematical statement of the control allocation problem.

Mathematically, a control allocator solves an underdetermined, typically constrained, system of equations. The input to the control allocator is the total control effect to be produced, the virtual control input $v(t) \in R^k$. The output of the control allocator is the true control input $u(t) \in R^m$, where $m>k$.

Given $v(t)$, we need to find $u(t)$ such that

$$g(u(t)) = v(t)$$

where g: $R^m \mapsto R^k$ is the mapping from actual control input to virtual control input in the system to be controlled.

Considering the case of a linear dynamic system in state-space form

Eq. (2-1)
$$\dot{x} = Ax + B_u u$$
$$y = Cx$$

$x \in R^n$ is the system state, $u \in R^m$ is the input of the control signals, and $A \in R^{n \times n}$, $B_u \in R^{n \times m}$. Assuming that B_u has ran $k<m$, it has a null space of dimension $m-k$ in which the control input can be perturbed without affecting $\dot{x}$. Thus, there are several redundancies that can be resolved using the control allocation technique.

Since B_u is rank deficient it can be factorized as:

Eq. (2-2)
$$B_u = B_v B$$

where $B_v \in R^{n \times k}$ and $B \in R^{k \times m}$ are both of rank k. Introducing the virtual control input as:

Eq. (2-3)
$$v = Bu$$

where $v \in R^k$ and B is known as the control effectiveness matrix. The system dynamics

can be rewritten as:

Eq. (2-4) $$\dot{x} = Ax + B_v v$$

$$y = Cx$$

To incorporate actuator force constraints, we require that:

Eq. (2-5) $$u_{\min,i} \leq u_i \leq u_{\max,i} \quad \text{for } i = 1,...,m$$

Given the limits, an exact solution may not exist, despite the redundancy. Furthermore, even if an exact solution exists, it cannot be assumed to be unique. Finding a solution to Eq. (2-3) within the constraints of Eq. (2-5) is defined as the control allocation problem.

For control of UAV, the state vector x can include the position, velocity, heading, roll and pitch. The output vector y may contain position, heading rate, roll rate and pitch rate. The control vector u contains forces generated by four propellers driven by four motors.

2.2. Fault Modeling of Control Surfaces

The objective of this section is to model faults so that the reconfigurable mechanism can be evaluated under different fault scenarios. A fault is any kind of malfunction which occurs in a system and that results in system instability and unacceptable performance degradation. Faults can occur in any component or part of the system/plant, such as actuator faults, sensor faults, structural or dynamic faults.

The control surfaces of an UAV can be affected by several types of faults. The control surfaces can be fully stuck at neutral/non-neutral position, or they can suffer a partial loss of the control surface area.

Common control surface faults to be considered in this thesis include [11]:

(1) Loss of effectiveness (LOE) fault,

(2) Freezing or lock-in-place (LIP) fault,

(3) Floating fault,

(4) Hard-over fault (HOF).

Loss of effectiveness is characterized by lowering the actuator gain with respect to its nominal value. In the case of LIP fault, the actuator freezes at a particular position and does not respond to subsequent commands. HOF is characterized by the actuator moving to its upper or lower limits regardless of the commanded signal. The speed of actuator response is bounded by the actuators rate limits. A floating fault occurs when the actuator floats with zero moment and does not contribute to the control effectors.

Different types of actuator faults may be mathematically parameterized as follows:

$$u_{true} = \begin{cases} u_{cmd} & No \quad Failure \quad Case \\ k(t)u_{cmd} & 0 < \varepsilon \le k(t) < 1, \forall t \ge t_F (LOE) \\ 0 & \forall t \ge t_F (Float) \\ u_{cmd}(t_F) & \forall t \ge t_F (LIP) \\ u_m \quad or \quad u_M & \forall t \ge t_F (HOF) \end{cases}$$

Where t_F denotes the time instant of fault occurrence in the actuator, k denotes its effectiveness coefficient such that $k \in [\varepsilon,1]$ and ε denotes its minimum effectiveness, u_m and u_M denote the minimum and maximum values of the input, respectively.

The above fault modeling can be represented in a general form as follows:

$$u_{true} = \sigma k u_{cmd} + (1-\sigma)\overline{u}$$

This includes all the above cases in a single representation, where u_{true} is the actuator

output, u_{cmd} is the output of the controller (also an input to the actuator), $\sigma = 1$ and $k = 1$ represent the fault-free case, $\sigma = 1$ and $\varepsilon < k < 1$ represent the cases of partial loss of control effectiveness fault. $\sigma = 0$ represents a float, LIP or HOF fault case. $u_m \leq \bar{u} \leq u_M$ is the position at which the actuator locks in the case of float, lock-in-place and hard-over faults.

For modeling the faults, it is assumed that the linearized dynamics of the normal or fault-free UAV at a trim condition is given by:

Eq. (2-6)
$$\dot{x} = Ax + B\,u$$
$$y = Cx$$

where $x \in R^n$ is the system state, $u \in R^m$ is the input of the control signals, $y \in R^p$ represents the output that will follow the command input, and $A \in R^{n \times n}$, $B \in R^{n \times m}$ and $C \in R^{p \times n}$ are matrices.

It is also assumed that the baseline flight control law and the control allocator have been designed to provide satisfactory stabilization and command tracking performance for the UAV under normal flight conditions.

2.2.1. Partial Loss

During normal operation, the actuator would operate exactly as directed by the controller. This means that the actuators are 100% effective in executing the control commands. When a partial loss occurs in a control actuator, the actuator would not be able to

completely track the control command given by the baseline controller. Therefore, the baseline control law or control allocator needs to be recalculated (reconfigured) based on the knowledge of the fault that occurred. One way to quantify the magnitude of the actuator fault is by defining a parameter known as control effectiveness factor γ [3]. The control effectiveness factor will represent the loss of the one to one relationship between the control command (controller output) and the true actuator action, as shown in Figure 2.3.

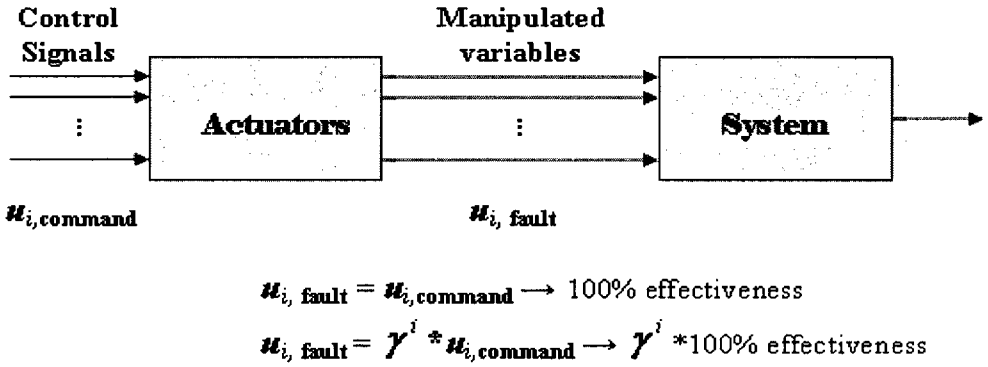

Figure 2. 3 Actuator failures by control effectiveness factor

where γ^i is the control effectiveness factor, $i = 1,...,m$

The system with the actuator faults modeled by control effectiveness factors can be written as:

Eq. (2-7)
$$\dot{x} = Ax + B_f u$$
$$y = Cx$$

where B_f is the post-fault control input matrix related to the nominal control input matrix

B. So, B_f can be modeled in the following way:

Eq. (2-8) $$B_f = B * \Gamma$$

where

$$\Gamma = \begin{bmatrix} \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \gamma^m \end{bmatrix}$$

γ^i=1, $i = 1,...,m$, denotes the healthy i^{th} actuator and γ^i=0 corresponds to total failure in the i^{th} actuator. Naturally, $0 < \gamma^i < 1$ represents partial loss in control effectiveness. For example, γ^1 =0.1 means the remaining effectiveness in the first control surface is 10%, or in other words, the control surface has lost 90% of its effectiveness.

Since the control effectiveness factors are parameters located between the controller command and the actuator actions, modeling can be viewed as multiplicative faults in nature.

2.2.2. Stuck Failure

Stuck fault is one serious fault scenario. Once stuck, the actuators can no longer respond to control commands. It is difficult to deal with stuck actuator faults because the remaining actuators must compensate for the effects of the failed actuators in the overall system. In practical situations, stuck faults are much more challenging and difficulty to be handled than partial faults. Here is the brief description [14].

During normal flight, the motion of a UAV can be described by Eq. (2-6). The closed-

loop system is stable under this condition, and its states can follow those of a reference model. In the presence of stuck actuators, it is necessary that the closed-loop system will remain stable and that the system states can still follow those of the reference model.

The reference model has the form

Eq. (2-9)
$$\dot{x}_m = A_m x_m + B\, r$$
$$y_m = C x_m$$

where $x_m \in R^n$ is the system state of the reference model, $y_m \in R^p$ represents the output of the reference model, $r \in R^m$ is the input of the reference model, and $A_m \in R^{n \times n}$ is the system matrix.

Under normal system operation, subtracting the reference model from Eq. (2-6), it follows that

Eq. (2-10)
$$\dot{e} = A_m e + [A - A_m]x + B[u - r]$$

where $e = x - x_m$, representing the state tracking error.

The control input u can be designed as:

Eq. (2-11)
$$u = -F_1 y + r$$

where F_1 is a constant matrix such that $BF_1C = [A - A_m]$.

The tracking error system, Eq. (2-10), then takes the following form:

Eq. (2-12)
$$\dot{e} = A_m e$$

By definition, the tracking error dynamics, Eq. (2-9), are stable. Once some actuators are stuck, the input to physical model u will no longer be effective in maintaining the model's path because the corresponding control signals are blocked by the stuck actuators.

Under this circumstance, the controller has to be reconfigured by adjusting the remaining (healthy) control signals such that the system can still follow the reference model.

With stuck actuators, the system Eq. (2-6) can be written as:

Eq. (2-13)
$$\begin{aligned} \dot{x} &= Ax + B_1 u_1 + \bar{B}_2 \bar{u}_2 \\ y &= Cx \end{aligned}$$

where $u_1 \in R^{m_1}$ represents the remaining control input. $\bar{u}_2 \in R^{m_2}$ is a constant vector that contains the values at which the actuators are stuck, and $B_1 \in R^{n \times m_1}$ and $\bar{B}_2 \in R^{n \times m_2}$ are the input distribution matrices with $m_1 + m_2 = m$.

Stuck actuators reduce the number of healthy control surfaces. Their effects can be viewed as additional constant disturbances imposed onto the system, which may drive the system away from the desired path. The closed-loop system stability may also be affected due to the loss of some control channels.

2.3. Control Allocation of ALTAV

According to the ALTAV non-linear model, three controller outputs, heading angular rate and pitch and roll angular rates, determine the desired action to three axes of the UAV movement. These three variables are used to produce the control signals to be sent to the four motors. Therefore, the control allocation block has three input variables $(\dot{\phi}, \dot{\theta}, \dot{\gamma})$ and four output variables F_1, F_2, F_3, F_4. The control effectiveness matrix B_b is picked from the rows and columns in the B-matrix of the state-space model describing the lineralized dynamics of the actuator motors in the ALTAV model.

For control allocation implementation, some approximations of the non-linear model must be made; these enable us to consider the control allocation problem in a much simpler and easier way/form.

First, we ignore actuator dynamics. It is assumed that the actuators are capable of moving indefinitely fast and without offset problems.

Second, we think of control surfaces as moment generators. The benefit is that the actuator has an exact force or position and a generated aerodynamic moment.

3. Control Reallocation Methods

The main idea of control reallocation is that once one or more control surfaces get stuck or partially lost during the flight, control reallocation methods should be able to use the redundancy of operable control surfaces to cancel the effects of the jammed and partial loss of the control surfaces and provide the same, or almost the same, desired control inputs [4, 5, 9, 22, 24, 25].

Let us now review the most common methods for control allocation in the literature. Many proposed "methods" correspond to different ways of computing the solution for a certain control allocation objective, rather than different objectives. In this thesis, the aim is to make a clear distinction, for each control reallocation method, between what the solution is searched for and how the solution can be computed numerically. Four algorithms have been implemented for control reallocation, which will be described in this chapter.

1. Cascaded Generalized Pseudo-Inverse (CGI)
2. Fixed-point algorithm (Fix)
3. Direct Control Allocation (DCA)
4. Weighted Least Squares (WLS)

3.1. Optimization Based Control Allocation

Optimization based methods rely on the following pragmatic interpretation of the control allocation problem. Given a virtual control command ν, determine a feasible control input

u such that $Bu=v$. This can be considered in the following way:

- If there are several solutions, pick the best one.
- If there is no solution, determine u such that Bu approximates v as much as possible.

Description of Method

As a measure of how "good" a solution or an approximation is, the lp norm is used. For a particular p, we will refer to this as lp optimal control allocation. The lp norm of a vector $u \in R^m$ is defined as:

Eq. (3-1)
$$\|u\|_p = \left(\sum_{i=1}^{m} |u_i|^p \right)^{\frac{1}{p}} \quad \text{for } 1 \le m \le \infty$$

The optimal control input is given by the solution to a two-step optimization problem.

Eq. (3-2)
$$u = \arg\min_{u=\Omega} \|W_u (u - u_d)\|_p$$

$$\Omega = \arg \min_{u\min \le u \le u\max} \|W_v (Bu - v)\|_p$$

where u_d is the desired control input and W_u and W_v are weighting matrices. The above equation should be interpreted as follows: Given Ω, the set of feasible control inputs that minimizes $Bu - v$ (weighted by Wv), pick the control input that minimizes $u - u_d$ (weighted by W_u).

3.2. Problem Statement

Let the linearized dynamics of the normal UAV or aircraft at a trim condition be given by

Eq. (3-3) $$\dot{x} = Ax + B\,u$$

When one or more control surfaces suddenly have partial loss of their effectiveness or become stuck at an unknown position, Eq. (3-3) becomes

Eq. (3-4) $$\dot{x} = Ax + B_f u_f + d$$

where B_f is the post-fault control effectiveness matrix. For partial loss, $d = 0$. For a stuck fault, attention needs to be paid so that u_f is the remaining control surface and d is the input to the UAV or aircraft caused by the stuck surfaces.

Let $y = C_y x$ be a selected p-dimensional controlled output vector to be used in defining the control allocation, then the derivative of y is,

Eq. (3-5) $$\dot{y} = C_y \dot{x}$$

Substituting Eq. (3-4) into the above equation, one gets

Eq. (3-6) $$\dot{y} = C_y Ax + C_y B_f u_f + C_y d$$

In general, the number of control surfaces is greater than or equal to the number of control output vectors. Relying on the reference model which represents the desired dynamics of the closed-loop system, the healthy UAV or aircraft would produce control input u_m if all of the control surfaces were normal. Then, the derivative of y_m is

Eq. (3-7) $$\dot{y}_m = C_y Ax + C_y B\,u_m$$

The objective of the control reallocation is to seek u_f for enabling Eq. (3-6) to be as close as possible to Eq. (3-7) and may be rewritten as

Eq. (3-8) $$C_y B_f u_f + C_y d = C_y B\, u_m$$

So the actual $\dot{y} = \dot{y}_m$, y will remain approximate to y_m. Such a u_f can be determined by minimization of the quadratic function.

Eq. (3-9)

$$\min_u J = \frac{1}{2}[(1-\varepsilon)(C_y B_f u_f + C_y d - C_y B u_m)^T Q\,(C_y B_f u_f + C_y d - C_y B u_m) + \varepsilon u_f^T Q_2 u_f]$$

Subject to

Eq. (3-10) $$u_{\min} \le u \le u_{\max}$$

where $u_{\min}$ and $u_{\max}$ are the smallest and biggest control inputs acting on the control surfaces and Q is a positive-definite matrix of appropriate dimension.

3.3. Cascade Generalized Pseudo-Inverse Method

Pseudo-Inverse as a reconfigurable control method of an active fault-tolerant control system consists of changing the feedback gain to complete the reconfiguration of the fault system. In fact, it completes fault-tolerant control through reconstructing flight control allocation. Pseudo-Inverse, in some control surfaces/actuators failure, can make use of the remaining fault-free control surfaces/actuators, in an appropriate linear combination, to reconfigure the control signals of control surfaces. Mathematically, the method is usually expressed as multiplying one pseudo-inverse array before the original input, thus its name. This method is widely used for control allocation. The main reason is its calculation and application, which are extremely easy. In the 1980s, many researchers conducted theoretical studies and simulations on the pseudo-inverse of the flight control

system. Also this method has executed flight tests on xBQM-106 UAV [29].

The pseudo-inverse method gives the solution by disregarding Eq. (3-10) constraints; a solution can be obtained by minimizing the quadratic function:

Eq. (3-11)

$$\min_u J = \frac{1}{2}[(C_y B_f u_f + C_y d - C_y B u_m)^T Q\,(C_y B_f u_f + C_y d - C_y B u_m)]$$

Suppose $C_y B_f = B_y$, $C_y d = d_y$ and $C_y B\, u_m = v_d$, called as virtual control.

The above equation is rewritten as

Eq. (3-12) $$\min_u J = \frac{1}{2}[(B_y u_f + d_y - v_d)^T Q\,(B_y u_f + d_y - v_d)]$$

Subject to

$$B_y u_f + d_y = v_d$$

An explicit solution can be obtained as:

Eq. (3-13) $$u_f = B_y^+ (v_d - d_y)$$

where

Eq. (3-14) $$B_y^+ = B_y^T (B_y B_y^T)^{-1}$$

and B_y^+ is the pseudo-inverse of B_y.

The solution of the pseudo-inverse in Eq. (3-13) will not be feasible for all achievable virtual control inputs *v*, and several ways to accommodate this constrain have been proposed. The simplest alternative is to truncate Eq. (3-13) by clipping those components that violate some constraint. However, since this typically causes only a few control inputs to saturate, it seems natural to use the remaining control inputs to make up the difference.

Virnig and Bodson [9] propose a Redistributed Pseudo Inverse (RPI) scheme, in which the control inputs that violate their bounds in the pseudo-inverse solution are saturated and removed from the optimization. Then, the control allocation problem is resolved with only the remaining control inputs as free variables. Bordignon [22] proposes an iterative variant of RPI. Instead of only redistributing the control effect once, the author proposes to keep redistributing the inputs as they become saturated. This is known as the Cascaded Generalized Inverse (CGI) approach.

The method of CGI arises from the idea that if a generalized inverse commands a control to exceed a position limit, then that control should be set at the exceeded limit, and the rest of the controls redistributed to achieve the desired moment. This procedure can be used with either pseudo-inverse or generalized inverse weighted with a diagonal matrix.

Initially, a generalized inverse computed using Eq. (3-13) and Eq. (3-14) is used to allocate the controls given in response to some desired moment.

If none of the elements in the solution are saturated, then the desired moment lies within the limits of the constraints. If any of the elements in the solution exceed their constraints, that element is set equal to its constraint, and its effects at saturation are subtracted from the desired moment. The effect of a saturated control is equivalent to the control position multiplied by the column of B matrix which corresponds to that control. The resulting moment is the part of the moment demand that must be satisfied by the remaining control authority which is denoted as v. For example, if the i^{th} control saturates, then

Eq. (3-15) $$u_i = u_{i(sat)}, v - B_i u_{i(sat)}$$

Next, the saturated controls are removed from the problem. When a pseudo-inverse is used, this is done by removing the corresponding column, B_i, from B. The reduced B

matrix is denoted by B^*. The new pseudo-inverse is then computed by plugging B^* into Eq. (3-14) to get B^{*+}. Now the new solution is once again checked for saturation. If there is a saturated element, the algorithm runs one more time according to the above method. Ultimately, either no new control will be saturated, or all the remaining controls are saturated, or the reduced B will have n or fewer columns. When no new controls are saturated, an admissible solution is found. If all the controls are saturated, the controls are set to their limits and the moment is unattainable using this method.

In the following, an example is given to demonstrate the concept of CGI.

CGI Example

Suppose

$$u_1 + 2u_2 = v$$

$$B = [1 \quad 2],\ u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},\ v = 3.5$$

Constrained by

$$0 \le u_1 \le 2$$

$$0 \le u_2 \le 1$$

The initial values for u_d are given by

$$u_d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

First iteration, calculating Pseudo-inverse B^+:

$$B^+ = [1 \quad 2]^T \cdot ([1 \quad 2] \cdot [1 \quad 2]^T)^{-1}$$

$$= \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$

Then,

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = B^+ \cdot v = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix} \cdot 3.5 = \begin{bmatrix} 0.7 \\ 1.4 \end{bmatrix}$$

u_2 is infeasible since it exceed its limit at u_2=1. So, the control allocation problem is resolved with only u_1 as a free variable. Replacing the original B matrix by $B^* = [1 \quad 0]$, the virtual control input that should be produced by u_1 is given by

$$v^* = 3.5 - [1*0 + 2*u_2] = 3.5 - 2 = 1.5$$

Then, we need to operate the second iteration.

The solution is then given by

$$u_1 = B^{*+} \cdot v* = 1*1.5 = 1.5.$$

So, we get

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 1 \end{bmatrix}$$

Now, let us check whether this solution is feasible or not. v= 1.5+2*1=3.5 fits and the algorithm stops. In this example, Cascade Generalized Inverse (CGI) was successful since the output is the true solution after two iterations.

However if the constraints in the above example were changed to

$$1 \le u_1 \le 2$$

$$0 \le u_2 \le 1$$

Running the same procedures, after the first iteration, $u1 = 0.7$ and $u2$ = 1.4. These values exceed their constraints. According to CGI regulations, u1 and $u2$ are set to 1.

Checking v=1*1+2*1=3, this result is not satisfactory. So, this is an incorrect result. Using CGI does not guaranteed that the optimal solution will be found.

3.4. Fixed-Point Method

The fixed-point method is simple. In comparison with CGI, this method is used to handle actuator saturation [5]. Many of the computations need to be performed only once before iterations start. Remarkably, the algorithm also provides an exact solution to the optimization problem, and it is guaranteed to converge. Its drawback is that convergence of the algorithm can be very slow and strongly dependant on the problem. The number of iterations required can vary by orders of magnitude depending on the desired vector. In addition, the choice of the parameter ε is sensitive, as it affects the objectives, as well as the convergence of the algorithm [9]. The fixed-point method is based on the mixed allocation problem.

The fixed-point method finds the control input vector u by minimizing the quadratic function

Eq. (3-16) $$\min_u J = \frac{1}{2}[(1-\varepsilon)(B_y u_f + d_y - v_d)^T Q_1 (B_y u_f + d_y - v_d) + \varepsilon u_f^T Q_2 u_f]$$

For $$0 < \varepsilon < 1$$

For the case when the constraints of Eq. (3-10) were not considered, solving the QP problem becomes much easier and the solution u_f is obtained from the unique solution of the linear algebraic system $\frac{\partial J}{\partial u_f} = 0$, which gives

$$u_f = (1-\varepsilon)[(B_y^T Q_1 B_y + \varepsilon Q_2]^{-1} B_y^T Q_1 (v_d u - d_y)$$

For another case, Eq. (3-10) constraints are active and involved in the applications. Applying this method to the quadratic programming problem, the fixed-point algorithm is given as

Eq. (3-17) $$u_{k+1} = sat[(1-\varepsilon)nB_y^T (v_d - d_y) - (nM - I)u_k]$$

$$= [F(v_d - d_y) - Gu_k] \qquad k = 0,...,N-1$$

where *N-1* is the maximum number of iterations,

Eq. (3-18) $$M = (1-\varepsilon)B_y^T Q_1 B_y + \varepsilon Q_2$$

and

Eq. (3-19) $$n = \frac{1}{\|M\|_2}$$

Sat (·) is the saturation function that clips the components of the vector u to their allowable value.

$$sat_i(u) = \begin{cases} \underline{u_i} & u_i < \underline{u_i} \\ u_i & \underline{u_i} < u_i < \overline{u_i} \\ \overline{u_i} & u_i > \overline{u_i} \end{cases} \qquad i = 1,...,m$$

This algorithm provides an exact solution to the optimization problem, and is guaranteed to converge. The convergence can be very slow and choosing a proper value ε is very important. Whether ε is good or not directly affects the optimal solution. A large value speeds up the convergence but makes it hard for the algorithm to find the exact solution. A small value for ε leads to slightly slower convergence but the algorithm converges closer to its optimal solution.

The fixed-point algorithm can be interpreted as a gradient search method where the

iterations are clipped to satisfy the constraints.

An example

Consider the following.

$$B = [2 \quad 1] \, , V = 3$$

$$u_{\min} = [-1 \quad -1]^T$$

$$u_{\max} = [1 \quad 1]^T$$

$$\varepsilon = 0.001$$

Here, Q_1 and Q_2 are unit matrices and $d_y = 0$.

The first thing to find is the initial condition for **u**. This is done by

$$u = \frac{(u_{\min} + u_{\max})}{2} \Rightarrow \frac{\left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)}{2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To compute the output we use:

$$u_{k+1} = (Fv - Gu_k)$$

where

$$Fv = (1-\varepsilon) \cdot n \cdot B^T \cdot v$$

$$= (1 - 0.001) \cdot 0.2002 \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} \cdot 3 = \begin{bmatrix} 1.1998 \\ 0.5999 \end{bmatrix}$$

and:

$$G = n * M - I$$

First, we calculate M:

$$M = (1-\varepsilon)B^T B + \varepsilon I$$

$$= (1-0.001)([2 \quad 1]^T[2 \quad 1]) + 0.001\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$= \begin{bmatrix}3.9970 & 1.9980\\ 1.9980 & 1.0000\end{bmatrix}$$

Then, calculating n,

$$n = \frac{1}{\|M\|_2} = \frac{1}{\left\|\begin{bmatrix}3.9980 & 1.9980\\ 1.9980 & 1.0000\end{bmatrix}\right\|_2} = 0.2002$$

So,
$$G = 0.2002 \cdot \begin{bmatrix}3.9970 & 1.9980\\ 1.9980 & 1.0000\end{bmatrix} - \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$= \begin{bmatrix}-0.2000 & 0.3999\\ 0.3999 & -0.7998\end{bmatrix}$$

Inserting the above F and G matrices into Eq. (3-17) gives:

$$u_1 = \begin{bmatrix}1.1998\\ 0.5999\end{bmatrix} - \begin{bmatrix}3.9970 & 1.9980\\ 1.9980 & 1.0000\end{bmatrix}\begin{bmatrix}0\\ 0\end{bmatrix}$$

$$= \begin{bmatrix}1.1998\\ 0.5999\end{bmatrix}$$

The next thing is to check if any element in u_1 exceeded the saturation limit. This is done by:

$$sat_i(u) = \begin{cases} \underline{u_i} & u_i < \underline{u_i} \\ u_i & \underline{u_i} < u_i < \overline{u_i} \\ \overline{u_i} & u_i > \overline{u_i} \end{cases} \qquad i=1, 2$$

If one of the outputs exceeds the constraints it will be set equal to the constraint.

u_1 is given from the first iteration

$$u_1 = \begin{bmatrix} 1.0000 \\ 0.5999 \end{bmatrix}$$

Now, we are ready to do the next iteration. In the following table the results of the iteration are given with the initial guess of $u_0 = 0$.

Table 3.1: Results from each iteration

Iteration no.	u_1	u_2	error for u_1	error for u_2
0	0.0000	0.0000	1.0000	1.0000
1	1.0000	0.5999	0.000	0.4001
2	1.0000	0.6798	0.000	0.3202
3	1.0000	0.7437	0.000	0.2563
4	1.0000	0.7948	0.000	0.2052
5	1.0000	0.8357	0.000	0.1643
6	1.0000	0.8683	0.000	0.1317
7	1.0000	0.8945	0.000	0.1055
8	1.0000	0.9154	0.000	0.0846
9	1.0000	0.9321	0.000	0.0679
10	1.0000	0.9455	0.000	0.0545

To get a satisfactory solution, it is necessary to do more iteration. From this example, it can be seen clearly that u can converge to the optimal solution after repeated iterations. In this calculated example, the correct solution was achieved after 98 iterations.

The advantage of the CGI method is its numerical simplicity, but clipping must be applied to enforce the actual constraint and can yield poor results. It does not guarantee

that the optimal solution will be found. Fix method provides an exact solution to the optimization problem, but this algorithm is relative slow and requires relatively high number of iterations before a solution can be found.

3.5. Direct Control Allocation Method

The objective of direct control allocation is to find a control vector u that gives the best approximation of v in the given direction. Thus, direct control allocation weighs directionality over moment generation, which is an important characteristic especially for applications such as flight control. In a special case of matrix B, direct allocation provides a unique solution to the problem. The condition for this property is that any q rows of B must be linearly independent, where q is the number of rows in B [9]. For flight control, the number of rows in B is usually three. In this case, the three components of v in the model reference control law are the accelerations in p, q and r, as outputs are three rotational accelerations. The columns of B represent the contributions of the various control surfaces to each of the three rotational accelerations.

Given a matrix B, find a real number a and a vector u_i such that $J = a$ is maximized, subject to

Eq. (3-20) $$(B)u_i = av$$

Eq. (3-21) $$a = \frac{(B)u_i}{v}$$

and

$$u_{\min,i} \leq u_i \leq u_{\max,i}$$

If a>1, let

Eq. (3-22) $$u = \frac{u_i}{a}$$

Otherwise, let

$$u = u_i$$

An advantage of direct allocation includes the straight forwardness of the allocation problem. No design variables must be selected, since the solution to the problem is determined by the control effectiveness matrix (B) and the constraints. When a>1, no element in u will be saturated. A method of implementing direct allocation is to use linear programming.

3.5.1. Constrained Optimization Using Linear Programming

Linear programming (LP) is regarded as a practical application of mathematics, as its applications are broad and universal. LP is an optimization approach that deals with meeting a desired objective such as minimizing or maximizing cost, in the presence of linear constraints such as limited resources.

Standard form:

The basic linear programming problem consists of two major parts:

- The objective function, and
- A set of constraints

For a maximization problem, the objective function is expressed as:

$$\text{Max } Z = c_1x_1 + c_2x_2 + c_3x_3 + \ldots + c_nx_n$$

where cj = payoff of each unit of the j^{th} activity that is undertaken and xj = magnitude of the j^{th} activity.

The constraints can be considered as:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &\le b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{21n}x_n &\le b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n &\le b_m \end{aligned}$$

Also expressed as

Eq. (3-23) $$Ax \le b$$

where $A \in R^{m \times n}$, aij = amount of the i^{th} resource that is consumed for each unit of the j^{th} activity and $b \in R^m$, bi = amount of the i^{th} resource that is available. That is, the resource is limited. The second general type of constraint specifies that all activities must have a positive value.

$$x_1 \ge 0,\ x_2 \ge 0, \cdots, x_n \ge 0$$

Simplified as

Eq. (3-24) $$x \ge 0$$

Together, the objective function Eq. (3-23) and the constraints Eq. (3-24) specify the linear programming problem.

3.5.2. Linear Programming

Bodson re-formulated direct control allocation as a linear programming problem [9]. LP problem can be derived. When re-defining the control allocation problem to fit a linear programming formulation, the standard linear LP problem consists in finding a vector x which minimizes:

Eq. (3-25) $$J = c^T x$$

Subject to:

Eq. (3-26) $$0 \le x \le h,\ Ax = b$$

In this equation alternative formulations exist, replacing $0 \le x \le h$ by $x \ge 0$ and $Ax=b$ by $Ax \ge b$. However, these differences are not significant and do not effect our discussion. To solve a linear programming problem in its standard form from the control allocation problem, a matrix M must be defined. The largest element of v must be identified beforehand. The largest element in v is denoted v_{max}, while the two remaining elements of v are defined as v_1 and v_2. According to the position of the largest element in v, M is defined. The index of M corresponds to the position of the largest element in v. The matrix M is then defined as one of three cases:

$$M_3 = \begin{bmatrix} -v_{\max} & 0 & v_1 \\ 0 & -v_{\max} & v_2 \end{bmatrix},\quad M_2 = \begin{bmatrix} -v_{\max} & v_1 & 0 \\ 0 & v_3 & -v_{\max} \end{bmatrix}$$

$$M_1 = \begin{bmatrix} v_2 & -v_{\max} & 0 \\ v_3 & 0 & -v_{\max} \end{bmatrix}$$

Using this M matrix, we can define the LP problem in standard form. To solve the problem, A is first defined:

Eq. (3-27) $$A = M \cdot B$$

Then, b is defined as

Eq. (3-28) $$b = -A \cdot x_{\min}$$

Proceeding to define h, we have:

Eq. (3-29) $$h = x_{\max} - x_{\min}$$

The objective function (c^T) must also be defined according to the problem. We define c^T as:

Eq. (3-30) $$c^T = -B^T v$$

The equations are then set up in a standard LP tableau, and the linear programming problem is then solved. In our implementation, the MATLAB function "linprog" is used in program codes. Sometimes, the solution vector (x) must be scaled according to a scaling factor (a). According to the value of the scaling factor, a logical choice is made to determine whether or not the solution vector should be scaled. If the scaling factor is larger than one, the solution vector should be scaled.

The scaling factor is calculated as

Eq. (3-31) $$a = \frac{(Bu)^T v}{(v \cdot v^T)}$$

An example

We have:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 9 \\ 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

where $$\begin{bmatrix} -5 \\ -10 \\ -2 \\ -1 \end{bmatrix} \le u \le \begin{bmatrix} 5 \\ 10 \\ 2 \\ 1 \end{bmatrix}$$

We proceed by defining M. Since the largest element of v is v_2,

M is defined as

$$M = \begin{bmatrix} -9 & 0 & 0 \\ 0 & 0 & -9 \end{bmatrix}$$

Following the procedure described above and using Eq. (3-27), we define A:

$$A = \begin{bmatrix} -9 & 0 & 0 \\ 0 & 0 & -9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Using Eq. (3-28) we define b:

$$b = \begin{bmatrix} -9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -5 \\ -10 \\ -2 \\ -1 \end{bmatrix} = \begin{bmatrix} -45 \\ -27 \end{bmatrix}$$

Using Eq. (3-29) we define h:

$$h = \begin{bmatrix} 5 \\ 10 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} -5 \\ -10 \\ -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 4 \\ 2 \end{bmatrix}$$

Using Eq. (3-30) we define the objective function (C_T):

$$c_T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 9 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -9 \\ 0 \\ -9 \end{bmatrix}$$

Writing the linear programming tableau, we define the following:

	X_1	X_2	X_3	X_4	b
c	0	-9	0	-9	0
R1	-9	0	0	0	-45
R2	0	0	-9	-9	-27

Where row c is the objective function and *R1* and *R2* are the rows of A.

Looking at the objective function, it can be seen that we must increase X_2 and X_4 to obtain a better value of the objective function. To do this, both X_2 and X_4 are driven to their saturated values, $X_2 = 20$ and $X_4 = 2$.

We obtain the following tableau:

	X_1	X_3	b
c	0	0	0
R1	-9	0	-45
R2	0	-9	-9

This gives an easy solution for both X_1 and X_3: $X_1 = 5$ and $X_3 = 1$.

The x vector then becomes

$$x = \begin{bmatrix} 5 \\ 20 \\ 1 \\ 2 \end{bmatrix}$$

Before we arrive at the final solution, we must first return from the LP problem definition, and obtain a formulation for use with the control allocation problem.

Continuing to use formulation, we calculate:

$$u = x + x_{\min} = \begin{bmatrix} 0 \\ 10 \\ -1 \\ 1 \end{bmatrix}$$

At last, the scaling factor must be calculated and applied to the solution if appropriate. Using the following formula, the scaling factor is calculated:

$$a = \frac{(Bu)^T v}{(v \cdot v^T)} = \frac{99}{81} = 1.2222$$

Since $a>1$, all elements of u must be divided by a to complete the calculations. This gives a final solution of

$$u = \begin{bmatrix} 0 \\ 8.1918 \\ -0.8181 \\ 0.8181 \end{bmatrix}$$

In order to find out whether this solution produces the right moment in the right direction, we can calculate:

$$v = Bu = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 8.1918 \\ -0.8181 \\ 0.8181 \end{bmatrix} = \begin{bmatrix} 0 \\ 9 \\ 0 \end{bmatrix}$$

This calculation shows that the solution found using linear programming is correct.

3.6. Weighted Least Squares

In this section, we will consider the weighted least squares (WLS) method. WLS is one use of the active set methods [27, 28] to solve the l_2 optimal control allocation problem.

3.6.1. Active Set Method

Based on O. Härkegård [26], the active set method is widely used to solve constrained quadratic programming (QP), and it has been proven that an optimal solution can be achieved in a finite number of iterations. The use of the active set method has two obvious advantages:

1. Reduce the constraints of the question, thus enabling to solve the question easily.
2. Reduce the possibility of incompatibility with QP sub-problems.

The bound and equality constrained least squares problem may be written as follows:

$$\min_u \|Au-b\|$$

$$Bu = v$$

$$Cu \geq U$$

Here, $C = \begin{pmatrix} I \\ -I \end{pmatrix}$ and $U = \begin{pmatrix} \underline{u} \\ -\bar{u} \end{pmatrix}$, so $Cu \geq U$ is equivalent to $\underline{u} \leq u \leq \bar{u}$.

The active set method solves this problem by solving a series of equality constraint problems. The thinking is that, in each step, some of the inequality constraints are

regarded as equality constrains, and form the working set W, while the remaining inequality constraints are disregarded. The active set of the solution is the working set at optimum.

3.6.2. Weighted Least Squares Discussion

Considering the choice of norm, a common technique is to approximately reformulate the sequential optimization problem (3-2) as a weighted optimization problem:

$$u = \underset{\underline{u} \le u \le \bar{u}}{\arg\min} \left\| W_u (u - u_d) \right\|^p + \gamma \left\| W_v (Bu - v) \right\|^p$$

where $\gamma >> 1$ to emphasize that, primarily, $Bu - v$ should be minimized, choosing $p=2$.

When the control allocator is initiated, and there is no previous solution available, $u^0 = \frac{(\underline{u} + \bar{u})}{2}$ and $W = 0$ are selected.

The cost function is rewritten in standard form:

$$\left\| W_u (u - u_d) \right\|^2 + \gamma \left\| W_v (Bu - v) \right\|^2 = \left\| \begin{pmatrix} \gamma^{\frac{1}{2}} W_v v \\ W_u \end{pmatrix} u - \begin{pmatrix} \gamma^{\frac{1}{2}} W_v v \\ W_u u_d \end{pmatrix} \right\|^2$$

where

Eq. (3-32) $$A = \begin{pmatrix} \gamma^{\frac{1}{2}} W_v B \\ W_u \end{pmatrix}, \quad b = \begin{pmatrix} \gamma^{\frac{1}{2}} W_v v \\ W_u u_d \end{pmatrix}$$

Solving

Eq. (3-33) $$u = \arg\min_u \left\| Au - b \right\|$$

Eq. (3-34) $$Bu = v$$

Eq. (3-35) $Cu \geq U$

Let u^0 be a feasible starting point. A point is feasible if it satisfies Eq. (3-34) and Eq. (3-35). Let the working set W contain a subset of the active inequality constraints at u^0. For $i = 1, 2 \ldots$

Given a suboptimal iterate u^i, find the optimal perturbation p, considering the inequality constraints in the working set as equality constraints and disregarding the remaining inequality constraints. Solve

Eq. (3-36) $$\min_p \left\| A(u^i + p) - b \right\|$$

$$Bp = 0$$

$$p_i = 0, i \in W$$

For one situation, if $u^i + p$ is feasible, then set $u^{i+1} = u^i + p$ and compute the Lagrange multipliers in the form:

Eq. (3-37) $$A^T(Au - b) = (B^T \quad C_0^T)\begin{pmatrix} \mu \\ \lambda \end{pmatrix}$$

If all $\lambda \geq 0$, u^{i+1} is the optimal solution to Eq. (3-34). The iteration will stop with $u = u^{i+1}$. Otherwise, remove the constraints associated with the most negative λ from the working set.

For another situation, if $u^i + p$ is in feasible, need to determine the maximum step α length such that $u^{i+1} = u^i + \alpha p$ is feasible. Add the bounding constraint at u^{i+1} to the working set.

An example

Let us revisit the case using the CGI algorithm and see how the WLS algorithm solves this problem.

Same as the CGI algorithm example,

$$u_1 + 2u_2 = v$$

$$B = [1\ 2],\quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},\quad v = 3.5$$

$$1 \le u_1 \le 2$$

$$0 \le u_2 \le 1$$

$W_u = I$, $W_v = I$ and γ is big enough, chose $\gamma = 1 \times 10^6$. Set initial

$$u_0 = \frac{u_{\min} + u_{\max}}{2} = [0.5 \quad 1.5]^T$$

Calculating A and b according to Eq. (3-32),

$$A = \begin{pmatrix} \gamma^{\frac{1}{2}} W_v B \\ W_u \end{pmatrix} = \begin{pmatrix} 1000 & 2000 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} \gamma^{\frac{1}{2}} W_v v \\ W_u u_d \end{pmatrix} = \begin{pmatrix} 3500 \\ 0 \\ 0 \end{pmatrix}$$

Initial residual:

$$d = b - A * u = \begin{pmatrix} 1000 \\ -1.5 \\ -0.5 \end{pmatrix}$$

Perturbation:

$$p = A \setminus d = \begin{pmatrix} -0.8 \\ 0.9 \end{pmatrix}$$

Getting the new point:

$$u^1 = u^0 + p = \begin{pmatrix} 0.7 \\ 1.4 \end{pmatrix}$$

The solution of the new point is the same as the solution of the failure case using the CGI algorithm. The new point is not the optimized point; we need to compute distances to the different boundaries.

Since $\alpha < 1$ is the maximum step length, we initiate with one. Hence, the maximum step length α is solved for:

$$\alpha = 0.5556$$

So, updating the point and residual,

$$u^2 = u^1 + \alpha p = \begin{pmatrix} 1.0556 \\ 1.0000 \end{pmatrix}$$

$$d = b - A * \alpha p = \begin{pmatrix} 444.4447 \\ -1.0556 \\ -1.0000 \end{pmatrix}$$

Re-computing the optimal perturbation vector p,

$$p = A^* \setminus d = \begin{pmatrix} 0.4444 \\ 0 \end{pmatrix}$$

The new point is

$$u^3 = u^2 + p = \begin{pmatrix} 1.5 \\ 1.0 \end{pmatrix}$$

Calculating the λ by Eq. (3-37),

$$\lambda = 2.0$$

Since $\lambda \geq 0$, the optimal solution is found. This confirms that $u = u^3 = \begin{pmatrix} 1.0 \\ 1.5 \end{pmatrix}$ is the optimal solution to the problem.

4. ALTAV UAV Benchmark

The Almost-Light-Than-Air-Vehicle (ALTAV) simulation is designed to permit the examination and evaluation of candidate control algorithms for the operation of one or more such vehicles.

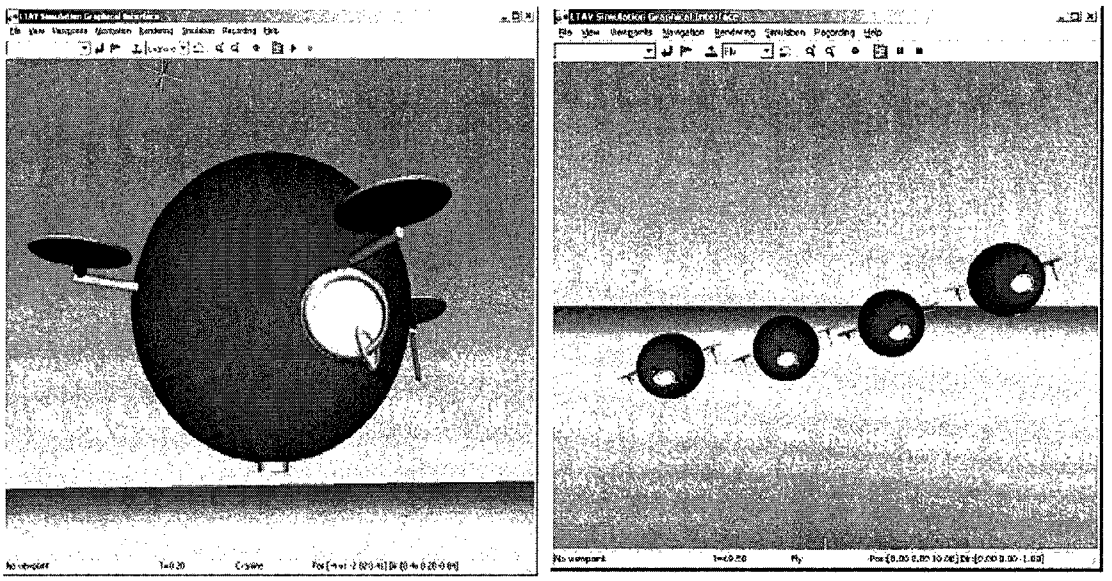

Figure 4. 1 ALTAV Simulink figure

The current implementation of the simulator software is designed primarily as a design and evaluation tool for the system state variable controllers. However, the simulation will also be suitable for testing higher level fleet control algorithms. It uses the Simulink Aerospace - Blockset Euler-6DOF block for tracking the vehicle's position and orientation through time [2].

4.1. ALTAV UAV Dynamics

The actual Almost-Light-Than-Aerial-Vehicle (ALTAV) system used in this paper is a six degrees of freedom unmanned aerial vehicle. The variables describing the motion are x, y, z, θ, γ and ϕ. These variables correspond to the translation in x, y and z directions and rotation about z, y and x axes (heading, pitch and roll), respectively. It should be noted that the system uses a 'right-hand' coordinate system with the positive z direction as down. The behavior of the ALTAV system is governed by the following equations (in the vehicle frame):

$$M\ddot{x} = F_i \sin(y) - C_x \dot{x}$$

$$M\ddot{y} = F_i \sin(\phi) - C_y \dot{y}$$

$$M\ddot{z} = -F_i s \cos(\gamma)\cos(\phi) - F_B + M_g - C_z \dot{z}$$

$$J_\theta \ddot{\theta} = (F_1 l - F_2 l + F_3 l - F_4 l)\sin(\rho) - C_\theta \dot{\theta}$$

$$J_\gamma \ddot{\gamma} = (F_1 l - F_3 l) - F_B L_B \sin(\gamma) - C_\gamma \dot{\gamma}$$

$$J_\phi \ddot{\phi} = -(F_2 l - F_4 l) - F_B L_B \sin(\phi) - C_\phi \dot{\phi}$$

where

M	Mass of the vehicle,
$x, y, z, \theta, \gamma, \phi$	Vehicle position and orientation,
$J_\phi, J_\gamma, J_\theta$	Moments of inertia about x, y and z axes, respectively,
F_B	Buoyant force resulting from the volume of helium in the vehicle,

$F_i, i = 1,...,4$ Force magnitude of motors,

l Perpendicular distance between the motors and vehicle center of gravity,

C_i Drag coefficient in the directions $i \in [x, y, z, \theta, \gamma, \phi]$ which serves as a damping term for the motor in that direction,

ρ Angular offset from vertical of the motor thrust vectors.

4.2. ALTAV Simulator Software

The ALTAV simulation is based on a physical model. The simulation software is basically composed of several files:

ALTAV_model.mdl

-Simulink model of ALTAV and the associated control and inputs for flying the vehicle.

-This model also includes a Simulink VR Toolbox graphical rendering subsystem (VR GUI block). This block is for generating a graphical interface only and has no effect on the simulation. In this paper, this block is out of our scope.

ALTAVSet.m

-Matlab script file which initializes the simulation with a square trajectory as the input path.

-Includes SetParams.m and SetPath.m.

ALTAVSetcircle.m

-Matlab script file which initializes the simulation when circle trajectory as input path.

-Includes SetParams.m and SetPathcircle.m.

SetParams.m

- Matlab script file which initializes all of the control, plant and physical parameters of the simulation.

SetPath.m/SetPathcircle.m

-These files assign the desired flight path trajectory for use by the system when not operating under manual control via a joystick. SetPath.m/SetPathcircle.m are two distinct trajectories are designed as model ALTAV input. The target trajectories are a square and a circle, respectively, which the vehicle attempts to follow.

Plot_compare.m

-Script file which executes the simulation for t time steps in order to gather performance information from the simulation.

Operating procedures:

- Starting Matlat
- Open ALTAV_model.mdl
- Execute ALTAVSet.m
- Start ALTAV_model.mdl simulation

4.3. Mdl Representation

The ALTAV_model.mdl diagram can be divided into four distinct regions (see Figure 4.2): the flight path input, the flight command controllers, the physical model of the system, and the "Real World" correction. The ALTAV simulation is wholly contained

within the 'ALTAV Simulation' with ALTAV_model.mdl.

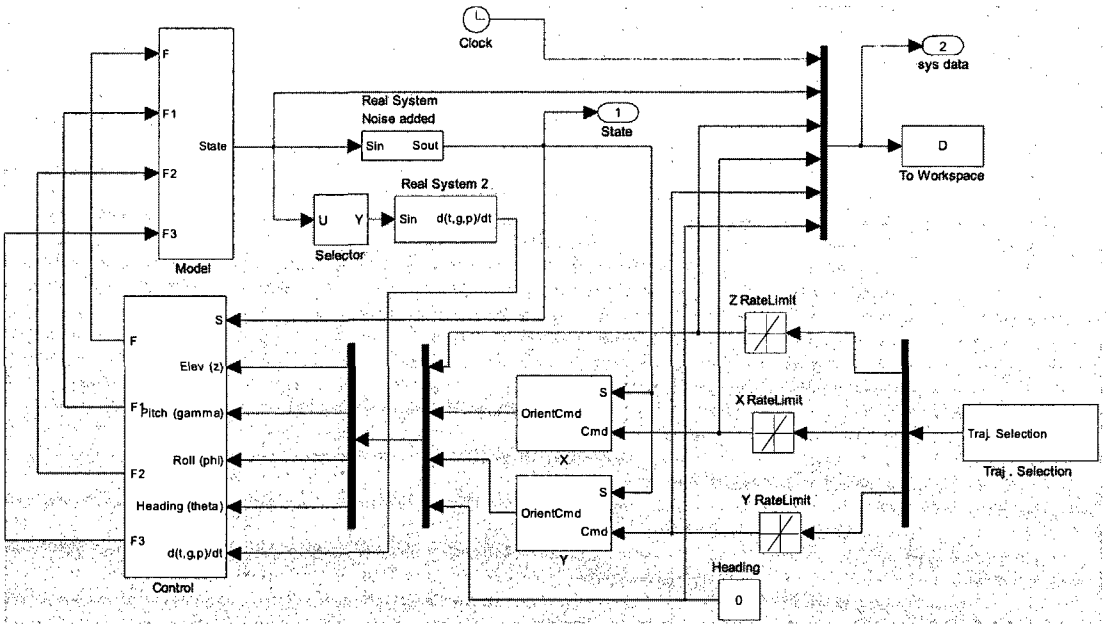

Figure 4. 2 Simulink block diagram of the ALTAV

4.4. Flight Path Input

The flight path block refers to as trajectory generators, generating a desired flight path for the vehicle, see Figure 4.3. The flight path is a sequence of time-stamped positions. Currently, the vehicle's flight paths are specified by (x, y) coordinates, elevation and vehicle heading. In this project, there are two different trajectory selection designs: squares trajectory and circle trajectory as model input. Square trajectory input is currently specified through the variables defined in SetPath.m, while circle trajectory input is designed through a Simulink block diagram. This information can be easily supplied from other sources, such as variables from the workspace or other generated flight paths.

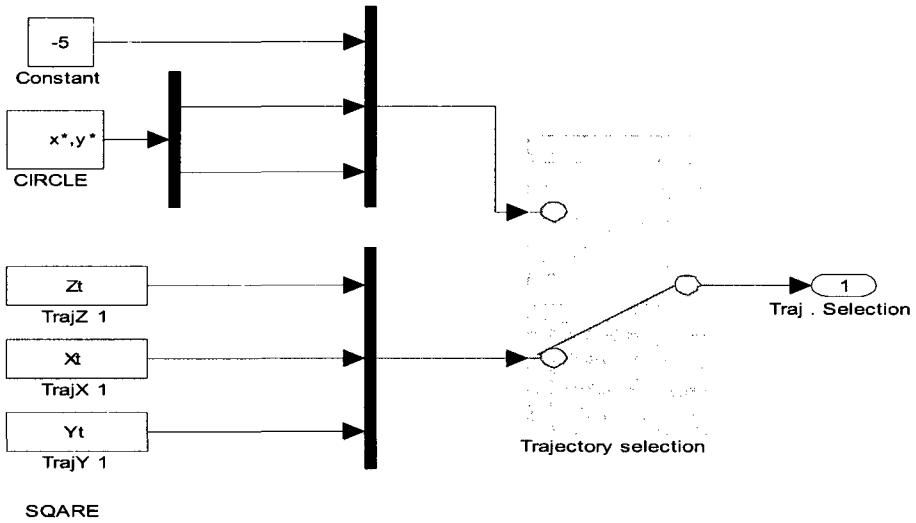

Figure 4. 3 ALTAV trajectory selections

In addition to the above mentioned variables for vehicle control is the 'V offset' input. This variable is used to investigate the behavior of the system when a vectored offset is added to the motors. This can be used in the future for improved handling in wind, though, at this stage, it is not used and the automatic control of the vehicle under such conditions is beyond the scope of this study.

4.5. Flight Command Controllers

Flight command controllers in the ALTAV UAV Simulink diagram have two levels. The first level of control generates pitch and roll command to move the vehicle in (x, y) space. In the next level, flight controllers provide commands to the four motors to maintain the specified elevation, yaw, pitch and roll. Within the 'Controller' block of the Simulink diagram, all controllers use a set of PID to accomplish control.

The specific parameters for controlling the vehicle are found in the Matlab script SetParas.m. Modeling of real vehicle characteristics is important to match the simulator to the real world as accurately as possible.

4.6. Real World Correction

In order to evaluate the performance of the system in the real world, it is necessary to model such effects as sensor noise and sensor delay. This block applies Gaussian noise to each of the system state variables with some predefined delay (as specified in SetParams.m). The noise is assumed to have zero mean and bias with a variance determined either theoretically or through experimentation.

In addition, each actuator has a rate limiter and a saturation cap to ensure that the system more closely mirrors the real system. These values have been selected based on empirical tests performed on the system components, such as maximum thrust and maximum vectoring rate. Again, these values can be seen and changed in SetParams.m.

4.7. Simulation Script Files

SetParams.m simulation parameters

The file SetParams.m contains all of the pertinent parameters for simulations. These include the parameters for the various PID controllers maintaining the state of the ALTAV to the physical specifications of the vehicles, which govern the vehicle flight characteristics.

The file is broken down as follows:

ALTAV Control Values

-These are the variables that specify the performance of the controllers running on the vehicle. The definition of variables is described in the comments part. The rate limits are used to prevent a change in target waypoint from being interpreted as a step input thus permitting much smoother operation.

Real System Values

-These are the values of the noise and sensor delays for the system sensors. If no noise is desired, the value of the variable NOISE is set to zero. Conversely, should noise be desired, NOISE should be set equal to one.

Motor Parameters

-This section defines the motor saturation values and the spin-up rate limits for the motors. In addition, the measured motor torque (as a function of generated thrust) can be included. This model assumes that all motors are rotating in the same direction (counter-clockwise).

Model System Physical Parameters

-This section defines such physical parameters as the mass of the system and the pertinent dimensional values. This includes different physical shapes for the vehicles and the appropriate configuration measurements.

It should be noted that the distances and coordinates referenced in this section are all relative to the CoG of the vehicle, rather than the centre of the vehicle itself. This protocol is used because the vehicle itself does not matter in the equations of motion, only the positions of the various components relative to the centre of gravity.

Hence, a Centre of Buoyancy (CoB) coordinate of [0, 0, 1] places the centre of buoyancy (usually the centre of volume) 1 meter directly above the centre of gravity and the vehicle is stable. [0, 0, -1] puts the CoB 1 m below the CoG for an unstable vehicle.

The Payload, Avionics Processor Unit (APU) and electronics and battery masses and positions can also be specified. This is relevant to both mass and lift calculations and moments of inertia calculations. This information is calculated here.

Model Initial Conditions

-This defines such initial conditions as the starting position, orientation and velocity of the vehicle. This is primarily used for testing and debugging.

5. ADMIRE Aircraft Benchmark

The ADMIRE (Aero-Data Model in Research Environment) benchmark has been developed from the aero data obtained from a generic, single seated, single engine fighter aircraft with a delta-canard configuration. The ADMIRE model is a non-linear, six degree of freedom simulation model of a small fighter aircraft consisting of a single engine delta-canard wing fighter aircraft model implemented in Matlab/Simulink and is maintained by the Department of Autonomous Systems of the Swedish Research Agency (FOI) [19, 20].

5.1. Aircraft Dynamic

The aircraft dynamics are modeled as a set of twelve first-order non-linear differential equations in the form:

Eq. (5-1) $$\dot{\bar{x}} = f(\bar{x}, \bar{u}, \Delta) \qquad \bar{x} \in \Re^{12\times 1},\ \bar{u} \in \Re^{12\times 1} \text{ and } \Delta \in \Re^{12\times 1}$$

Eq. (5-2) $$\bar{z} = g(\bar{x}, \bar{u}, \Delta) \qquad \bar{z} \in \Re^{31\times 1}$$

where $\bar{x}$ is the state vector, $\bar{u}$ is the input vector, $\bar{z}$ is the output vector, and Δ is the vector containing uncertain and variable parameters.

The ADMIRE flight operation region is up to mach 1.2 and altitude up to 6 km. The aircraft model has 12 state components related to aircraft dynamics $(V_T, \alpha, \beta, p_b, q_b, r_b, \Phi, \theta, \Psi, x_v, y_v, z_v)$ and additional components due to the presence of actuators and components of a flight control system. The force equations in the aircraft

body axes frame are:

Eq. (5-3) $$\dot{u} = \frac{F_{Xaero} + T}{m} - g\sin\theta - q_b w_b + r_b v_b$$

Eq. (5-4) $$\dot{v} = \frac{F_{Xaero}}{m} + g\cos\theta\sin\phi + p_b w_b - r_b u_b$$

Eq. (5-5) $$\dot{w} = \frac{F_{Zaero}}{m} + g\cos\theta\cos\phi - p_b v_b + q_b u_b$$

The components of state derivatives, vector $\bar{x}$ in equation (1), are as follows:

Eq. (5-6) $$\dot{V}_T = \frac{u_b \dot{u}_b + v_b \dot{v}_b + w_b \dot{w}_b}{V_T}$$

Eq. (5-7) $$\dot{\alpha} = \frac{u_b \dot{w}_b - w_b \dot{u}_b}{u_b^2 + w_b^2}$$

Eq. (5-8) $$\dot{\beta} = \frac{\dot{v}_b V_T - v_b \dot{V}_T}{V_T^2 \cos\beta}$$

Eq. (5-9) $$\dot{p} = \frac{(I_{xx} - I_{yy} + I_{zz})I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} p_b q_b + \frac{(I_{yy} - I_{zz})I_{zz} - I_{xz}^2}{I_{xx}I_{zz} - I_{xz}^2} q_b r_b + \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} M_z + \frac{I_{zz}}{I_{xx}I_{zz} - I_{xz}^2} M_x$$

Eq. (5-10) $$\dot{q} = \frac{-I_{xz}}{I_{yy}} p_b^2 + \frac{(I_{zz} - I_{xx})}{I_{yy}} p_b r_b + \frac{I_{xz}}{I_{yy}} r_b^2 + \frac{M_y}{I_{yy}}$$

Eq. (5-11) $$\dot{r}_b = \frac{(I_{xx} - I_{yy})I_{zz} + I_{xz}^2}{I_{xx}I_{zz} - I_{xz}^2} p_b r_b + \frac{(I_{yy} - I_{zz} - I_{xx})I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} q_b r_b + \frac{I_{xx}}{I_{xx}I_{zz} - I_{xz}^2} M_z + \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} M_X$$

Eq. (5-12) $$\dot{\psi} = \frac{q_b \sin\phi + r_b \cos\phi}{\cos\theta}$$

Eq. (5-13) $$\dot{\theta} = q_b \cos\phi - r_b \sin\phi$$

Eq. (5-14) $$\dot{\phi} = p_b + \tan\theta(q_b \sin\phi + r_b \cos\phi)$$

The output vector consists of the state variables, plus additional variables defined by the equations:

Eq. (5-15) $$u_b = V_T \cos\alpha \cos\beta$$

Eq. (5-16) $$v_b = V_T \sin\beta$$

Eq. (5-17) $$w_b = V_T \sin\alpha \cos\beta$$

The above three equations provide the body-axis velocities u_b, v_b and w_b, respectively. The engine thrust works in the direction of the x_b axis. So, load factors along the z_b and y_b axes are derived from the total aerodynamic forces in the body fixed z and y axes only.

Eq. (5-18) $$n_z = \frac{-F_{Zaero}}{mg_0}$$

Eq. (5-19) $$n_y = \frac{F_{Yaero}}{mg_0}$$

The Mach number M and flight angle γ are computed as:

Eq. (5-20) $$M = \frac{V_T}{a(h)}$$

Eq. (5-21) $$\gamma = -\arcsin\left(\frac{\dot{z}_v}{V_T}\right)$$

The coefficients of drag and lift, defined in the stability-axes frame are:

Eq. (5-22) $C_D = C_N \sin\alpha + C_T \cos\alpha$

Eq. (5-23) $C_L = C_N \cos\alpha - C_T \sin\alpha$

The side force coefficient C_Y, roll moment coefficient C_l, pitch moment coefficient C_m, yawing moment coefficient C_n and forces F_{Xaero} and F_{Zaero} in the x_b, y_b and z_b axis are generated as outputs.

5.2. ADMIRE Model

To evaluate the designed control allocation algorithms produced in this thesis, the ADMIRE model is used for simulation.

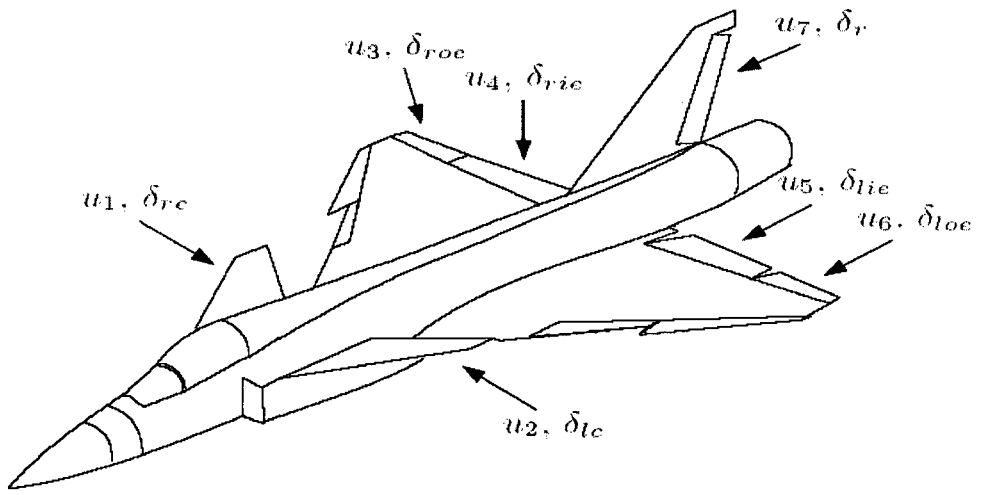

Figure 5. 1 ADMIRE control surface configurations

Further details about ADMIRE:

1. Dynamics: The dynamic model consists of the nonlinear rigid body equations, along with the corresponding equations for the position and orientation. Actuator

and sensor dynamics are included.

2. Aerodynamics: The aero-data model is based on the Generic Aero-data Model (GAM) developed by Saab AB and was extended for high angles of attack.
3. Control surfaces: The actuator suite consists of canards (left and right), leading-edge flaps (left and right), elevons (inner, outer, right and left), a rudder and thrust vectoring capabilities. In this model, the landing edge flaps will not be used for control allocation since these do not produce large aerodynamic moments. Thrust vectoring will also not be used in this project due to lacking documentation. The remaining seven control surfaces are denoted in Figure 5.1. u denotes the commanded deflection while δ represents the actual deflection.
4. Actuator models: The servo dynamics of the utilized control surfaces are given by first-order systems with a time constant of 0.05s, corresponding to a bandwidth of 20 rad/sec. Actuator position and rate constraints are also included. Table 1 shows the actual rate and position constraints for flight below Mach 0.5.
5. Flight envelope: The flight envelope covers Mach numbers up to 1.2 and altitudes up to 6000m. Longitudinal aero-data exist up to an angle of attack of 90 degrees, while lateral aero-data only exist for angles of attack up to 30 degrees.

ADMIRE control surface limits below Mach 0.5

Control surface	Min. deflection(deg)	Max deflection(deg)	Max. rate(deg/sec)
Canards	-55	25	50
Elevons	-30	30	150
Rudder	-30	30	100

5.3. ADMIRE Simulation Model

The ADMIRE simulation model includes four blocks: the Flight Control System block, Computer Delay and Transport Delay block, Saturators Rate-limiters and Actuators block and Aircraft Response block. The ADMIRE non-linear model in Simulink is represented as follows.

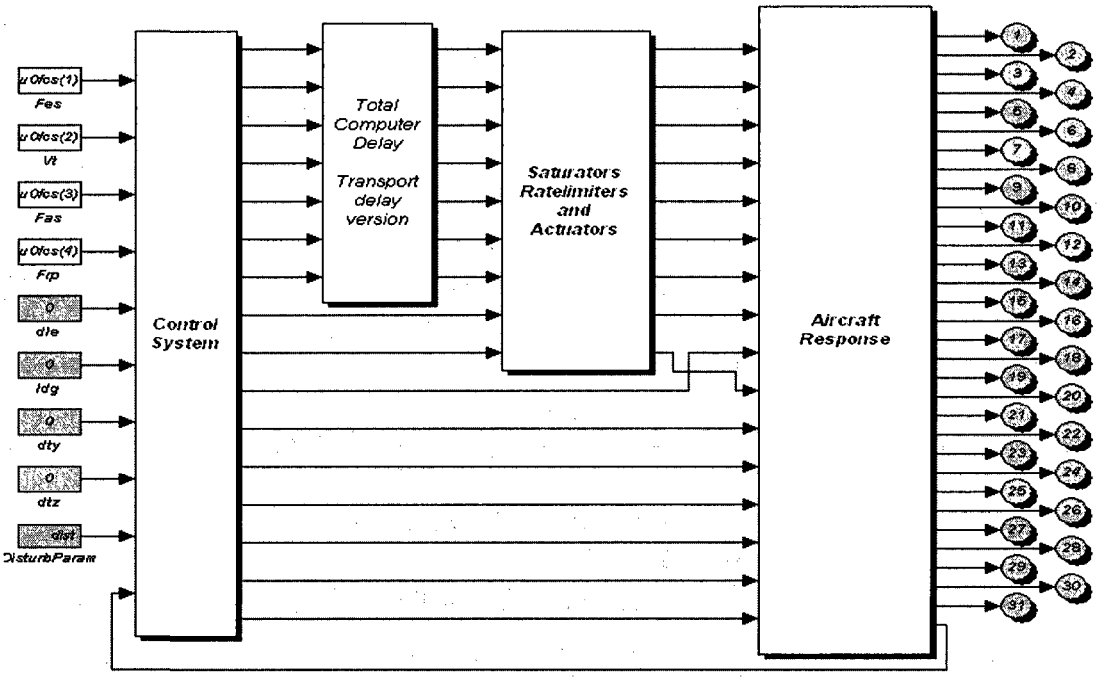

Figure 5. 2 Simulink block diagram of the ADMIRE

Figure 5.2 shows the Simulink block diagram of the ADMIRE benchmark. The fault model was not included since the original aircraft model was not implemented for fault-tolerant control search purposes. The inputs are given by the pilot, such as longitudinal (Fes) and lateral (Fas) stick deflection, rudder pedal deflection (Frp), and throttle stick setting (Tss).

6. Control Reallocation Implementation

This section will present the implementation of different control allocation methods in the ALTAV and ADMIRE Simulink nonlinear models. Simulations are conducted in order to analyze whether different control allocation algorithms possess the ability to re-stabilize UAV or aircraft and provide reasonable command-tracking performance. The following implementation has been considered:

1. Partial loss of the control input.
2. Stuck at unknown position.

This section is organized as follows. The implementation of the UAV ATLAV model is presented in 6.1. The implementation of the Aircraft ADMIRE model is presented in 6.2.

6.1. Implementation in ALTAV

The files for partial and stuck faults in the ALTAV model are differentiated as:

For partial loss:

ALTAV_model_test_partial_fault.mdl (Gaussian noise added),

ALTAV_model_test_partial_fault_simple.mdl (no Gaussian noise)

For stuck fault:

ALTAV_model_test_stuck_fault.mdl (Gaussian noise added)

ALTAV_model_test_stuck_fault_simple.mdl (no Gaussian noise)

6.1.1. Simulink Block for Nonlinear Model

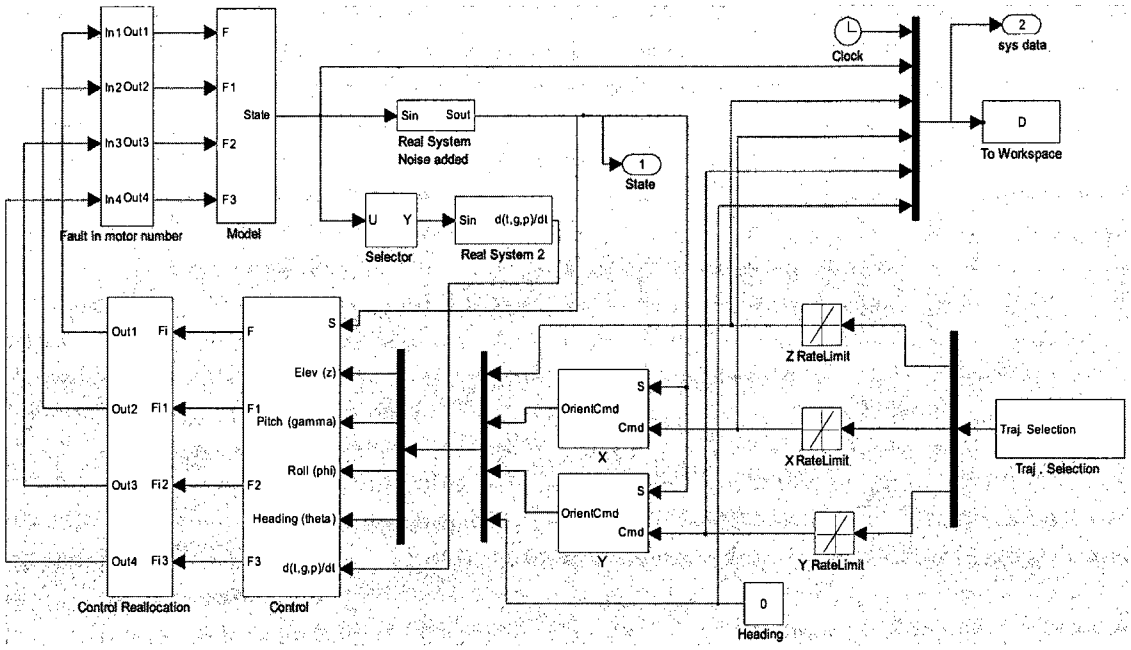

Figure 6. 1 ALTAV simulink with control reallocation block (Gaussian noise added)

The description for Figures 6.1 and 6.2 are given in Section 4.3, 4.4, 4.5 and 4.6.

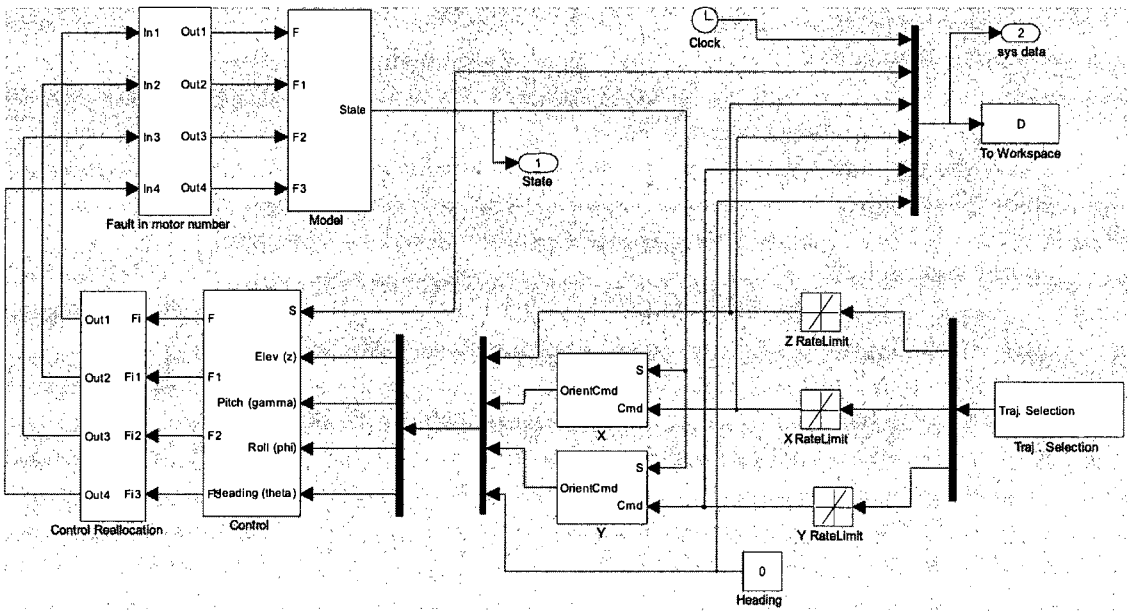

Figure 6. 2 ALATV Simulink with control reallocation block (no Gaussian noise)

6.1.2. Implementation of Partial Loss

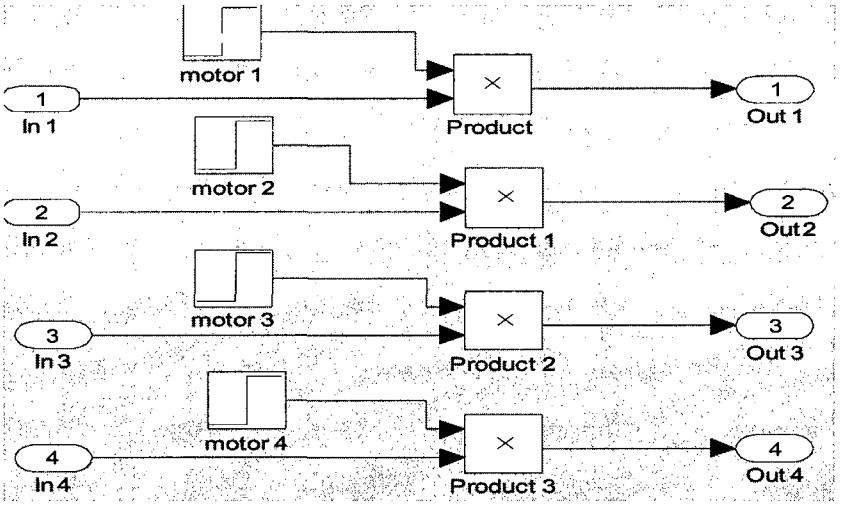

Figure 6. 3 ALTAV Simulink block of partial loss implementation

Figure 6.3 show faults in control surfaces implementation. The control effectiveness factor γ, described in section 2.2.1, is represented by Simulink step blocks which multiply the control motor signal. The parameter 'Final value' denotes the numerical value of γ and the parameter "Step time" sets the time when the fault occurs. In normal conditions, the parameter "Final value" must remain in 1 and the "Step time" must be zero. These values are given in Matlab prompt using the file *ALTAV_Bfault_partial.m* in ALTAV, *ADMIRE_Bfault_partial.m* in ADMIRE.

6.1.3.Implementation of Partial Loss

Stuck at non-zero position:

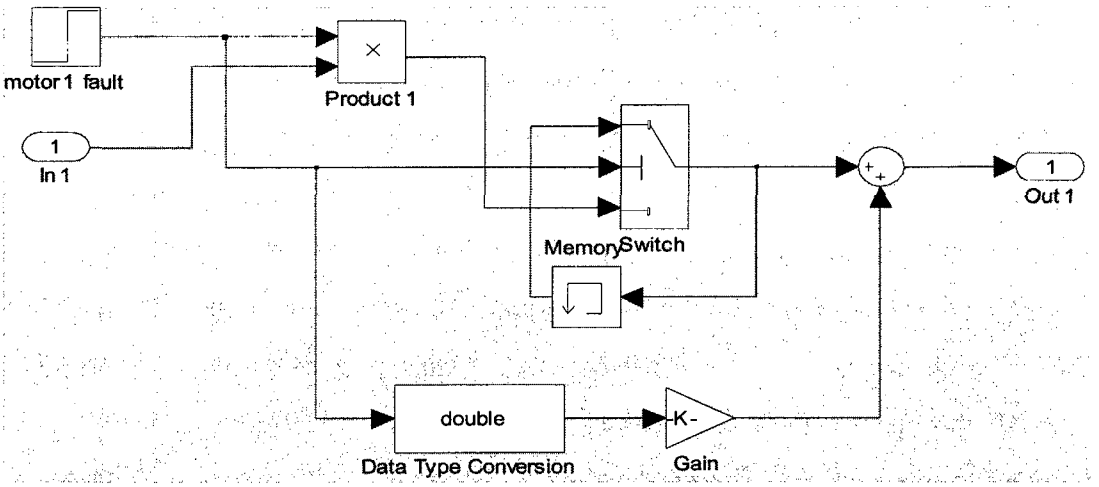

Figure 6. 4 Simulink block for stuck at current position

Figure 6.4 is used to generate the stuck fault at a non-zero position in the Simulink model. The gain (k) is the magnitude of the fault. This is implemented only in the motor1 control surface.

Runaway

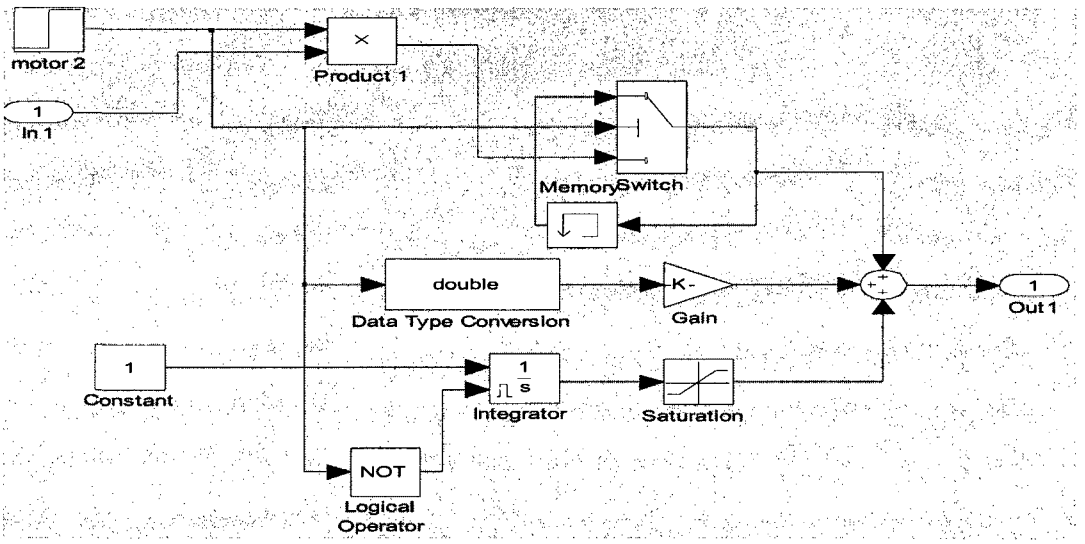

Figure 6. 5 Simulink block for runaway

Figure 6.5 shows the fault implementation of a runaway in the Simulink block. This is implemented only in the motor2 control surface. The faults are selected using the file ALTAV_Bfault_stuck.m in the Matlab prompt.

In the ALTAV non-linear model, the location of the fault is placed in the UAV response block, right before the system dynamics, as shown in Figures 6.1 and 6.2.

6.1.4. Implementation of Control Reallocation

The Control Re-allocation block is placed between the control system block and fault motor number block. Implementation of control re-allocation is divided into two parts according to the types of the UAV fault:

1) Partial loss

- The files that simulate the reconfigurable fault-tolerant control system are (Gaussian noise added)

  ALTAV_model_test_partial_CGI.mdl

  ALTAV_model_test_partial_Fixedpoint.mdl

  ALTAV_model_test_partial_DCA.mdl

  ALTAV_model_test_partial_WLS.mdl

- The files which simulate the reconfigurable fault-tolerant control system are (no Gaussian noise)

  ALTAV_model_test_partial_CGI_simple.mdl

  ALTAV_model_test_partial_Fixedpoint_simple.mdl

  ALTAV_model_test_partial_DCA_simple.mdl

ALTAV_model_test_partial_WLS_simple.mdl

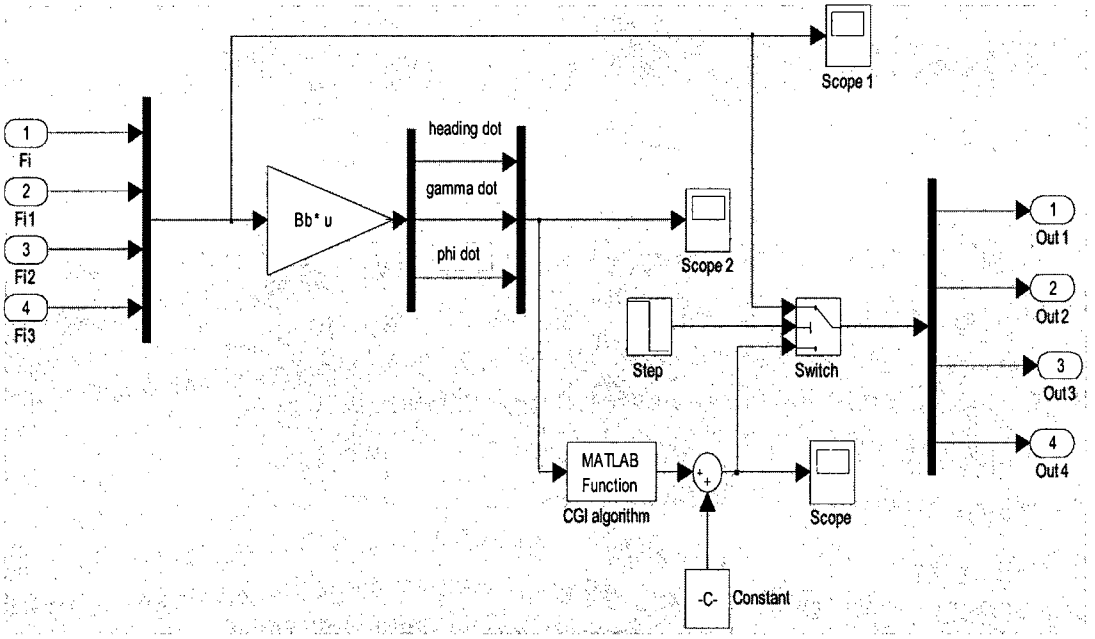

Figure 6. 6 ALTAV Simulink diagram of control reallocation as partial loss

Figure 6.6 shows control reallocation diagram when one or multiple control actuators have partial loss.

2) Stuck fault

- The files which simulate the reconfigurable fault-tolerant control system are (Gaussian noise added)

  ALTAV_model_test_stuck_CGI.mdl

  ALTAV_model_test_stuck_Fixedpoint.mdl

  ALTAV_model_test_stuck_DCA.mdl

  ALTAV_model_test_stuck_WLS.mdl

- The files which simulate the reconfigurable fault-tolerant control system are (no Gaussian noise)

ALTAV_model_test_stuck_CGI_simple.mdl

ALTAV_model_test_stuck_Fixedpoint_simple.mdl

ALTAV_model_test_stuck_DCA_simple.mdl

ALTAV_model_test_stuck_WLS_simple.mdl

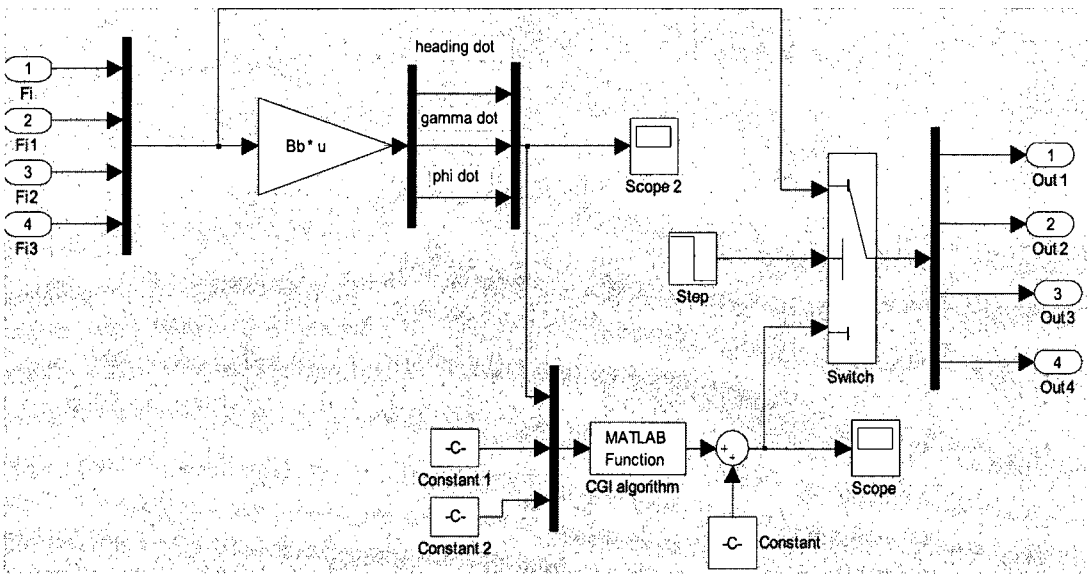

Figure 6. 7 ALTAV Simulink diagram of control reallocation as stuck fault

Figure 6.7 shows control reallocation diagram when one or multiple control actuators stuck at current position or unknown position.

6.2. Implementation in ADMIRE

The files for partial and stuck faults in the ADMIRE non-linear model are differentiated as

For partial loss:

ADMIRE_model_partial_fault.mdl

For stuck fault:

ADMIRE_model_stuck_fault.mdl

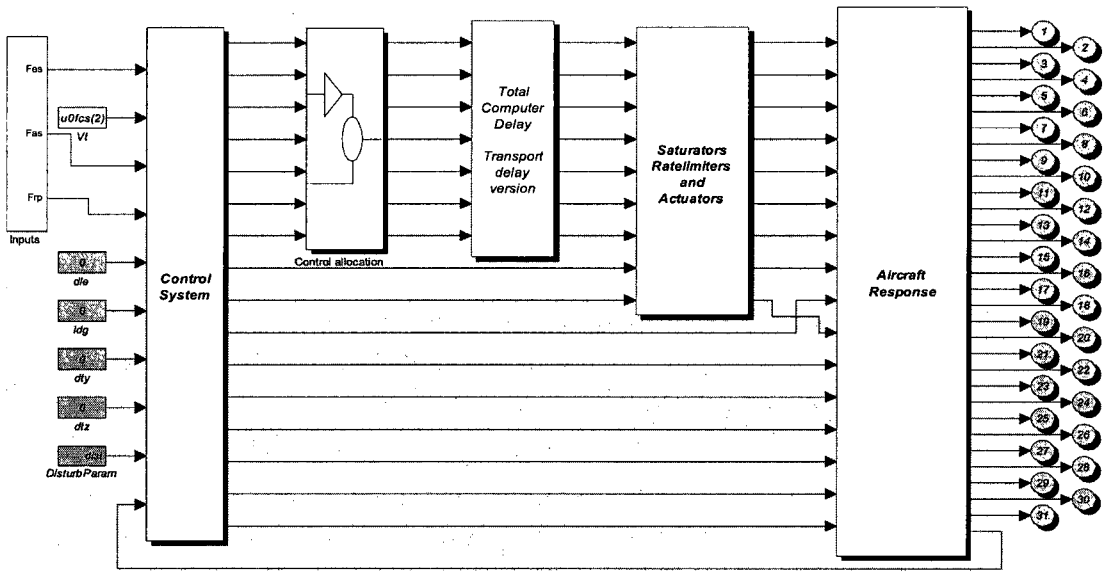

Figure 6. 8 ADMIRE Simulink with control reallocation block

In the ADMIRE non-linear model, implementation of partial loss and stuck fault is similar to the ALTAV non-linear model. It is worth noting that there are seven control surfaces in the ADMIRE aircraft model, while there are four in the ALTAV UAV model. Take partial loss as an example, shown in Figure 6.9.

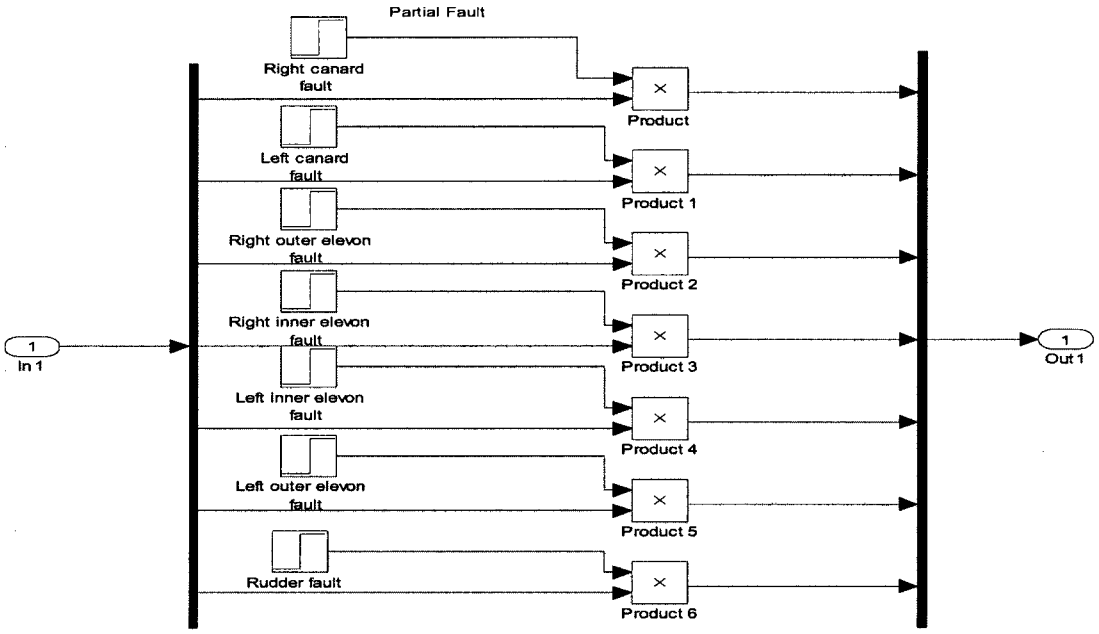

Figure 6. 9 ADMIRE Simulink block of partial loss implementation

Different Simulink files are given below in the ADMIRE non-linear model:

ADMIRE_model_partial_CGI.mdl

ADMIRE_model_partial_Fixedpoint.mdl

ADMIRE_model_partial_DCA.mdl

ADMIRE_model_partial_WLS.mdl

ADMIRE_model_stuck_CGI.mdl

ADMIRE_model_stuck_Fixedpoint.mdl

ADMIRE_model_stuck_DCA.mdl

ADMIRE_model_stuck_WLS.mdl

7. Simulation Results in ALTAV

In this chapter, as a part of the project, partial loss and stuck at an unknown position in the motor control surface and the reconfigurable control system are implemented in the ALTAV non-linear model. An evaluation of the influence of Gaussian noise is given first, then trajectory selection, and finally simulation results in the presence of partial loss and stuck failures are presented. Simulations are conducted in order to investigate the ability of the reconfigurable control system to re-stabilize the aircraft and provide reasonable command-tracking performance.

In the ALTAV non-linear model, there are two different trajectories as model control inputs. Simulation time varies according to the input trajectory:

1) Square trajectory: simulation time is 80s, fault is generated at 50s and the control reconfiguration starts at 50s.
2) Circle trajectory: simulation time is 120s, fault is generated at 80s and the control reconfiguration starts at 80s.

The simulation results shown below are compared to the four reconfigurable control algorithms, Pseudo-Inverse, Fixed Point, Direct Control Allocation and Weighted Least Squares algorithm. These algorithms generate the signals for deflection on the UAV motors.

7.1. Influence of Gaussian Noise

In order to evaluate the performance of the system in the real world, it is necessary to

model such effects as sensor noise and sensor delay. This block applies Gaussian noise to each of the system state variables with some predefined delay. The noise is assumed to have zero mean and bias, with a variance determined either theoretically or through experimentation. Two examples to explain:

(1) Square trajectory input, partial loss in motor 1 with 70%.

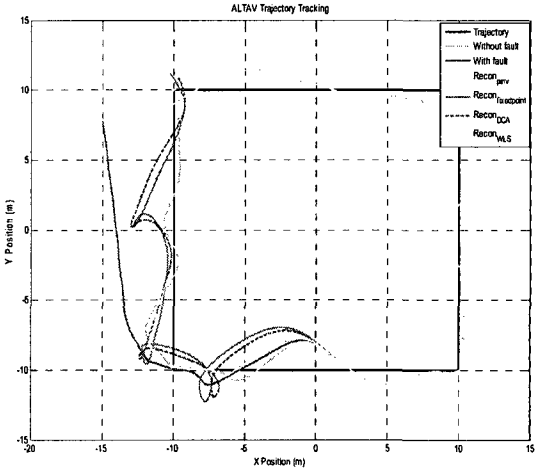

Figure 7. 1 Virtual and actual trajectory diagram (Gaussian noise added)

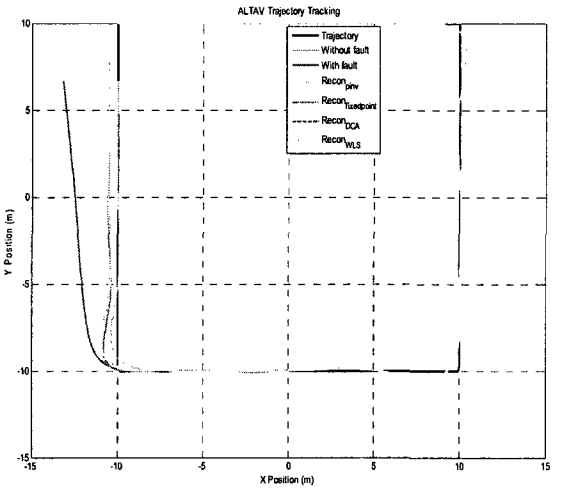

Figure 7. 2 Virtual and actual trajectory diagram (no Gaussian noise)

(2) Circle trajectory input, motor 3 stuck at 0.35

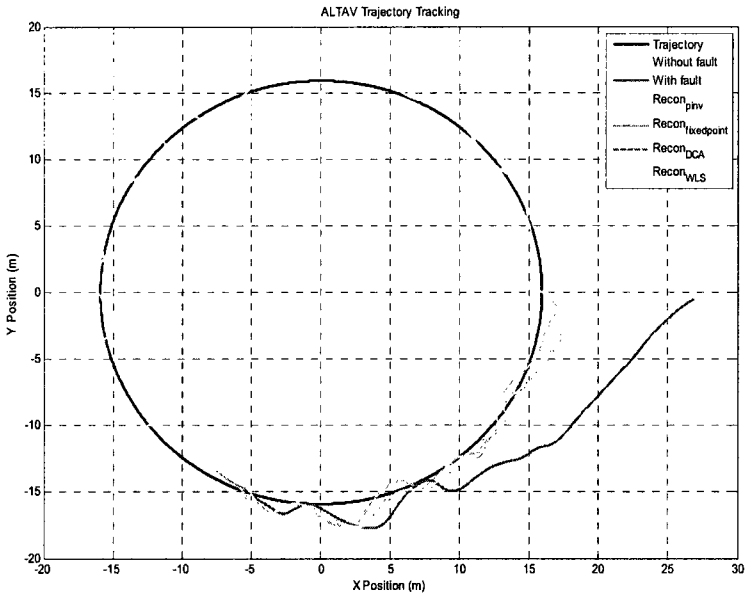

Figure 7. 3 Virtual and actual trajectory diagram (Gaussian noise added)

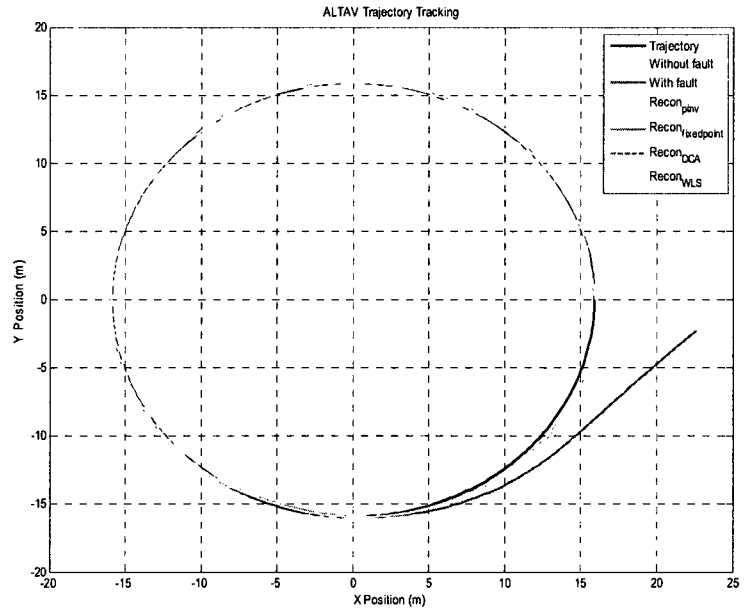

Figure 7. 4 Virtual and actual trajectory diagram (no Gaussian noise)

Figure 7.1 and 7.2 show the influence of with/without Gaussian noise to the model when the square trajectory is the input. Figure 7.3 and 7.4 show the difference between adding Gaussian noise and no Gaussian noise in the circle trajectory model. From these figures, we can see the CGI, Fix, DCA and WLS, their virtual trajectories with disturbance shaking around the desired trajectories, deflection error is bigger with Gaussian noise without, but the track is still following the square trajectory. In a practical system, different kinds of disturbances always exist, but in this paper, removing Gaussian noise will not affect the study of the problem.

Also, some reconfigurable models with Gaussian noise take time to operate, their calculations are long, and simulation is very slow. So, just the ALTAV Simulink models without Gaussian noise are considered in the following discussion.

7.2. Trajectory Selections

There are two different trajectories used as control input commands for the ALTAV platform. Let us compare the difference between these two trajectories.

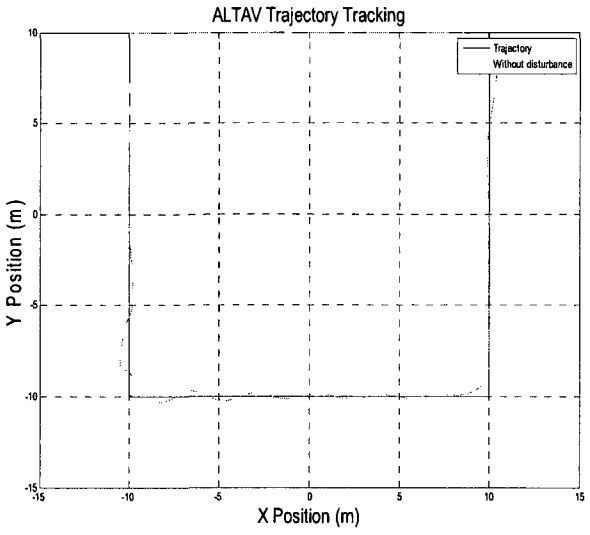

Figure 7. 5 The square virtual trajectory vs the desired trajectory

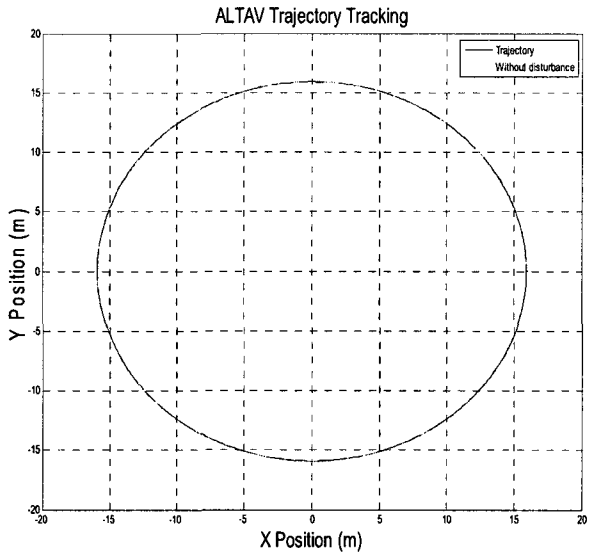

Figure 7. 6 The circle virtual trajectory vs the desired trajectory

Figure 7.6 shows, in the original ALTAV simulation model, the virtual trajectory without disturbance completely tracks the desired trajectory without any error under circle as commanded input. Figure 7.5 shows that the virtual trajectory follows the desired

trajectory with an error under square trajectory commanded input, its error magnitude increases when the trajectory suddenly changes the direction.

7.3. Simulation Results for Partial Loss

Partial loss control effectiveness scenario is considered as follows:

Input command: Square trajectory as model input

Scenario	Partial Loss
1	Motor 1 with 75 % loss
2	Motor 3 with 50 % loss
3	Motor 4 with 80 % loss

Input command: Circle trajectory as model input

Scenario	Partial Loss
1	Motor 2 with 80 % loss
2	Motor 3 with 90 % loss
3	Motor 4 with 75 % loss

For an easy comparison with all different control re-allocation algorithms, together with normal and faulty responses, the simulation result includes seven curves in each graph. These curves are:

Black: Trajectory

Green: Without fault

Blue plus solid line: Fault

Yellow: Reconfiguration with CGI method

Magenta: Reconfiguration with Fix method

Red plus dash line: Reconfiguration with DCA method

Cyan: Reconfiguration with WLS method

The following simulation results are related to partial loss without Gaussian noise.

7.3.1.Square trajectory as model input

Scenario 1:

Input: Square input

Output: motor 1 with 75% loss

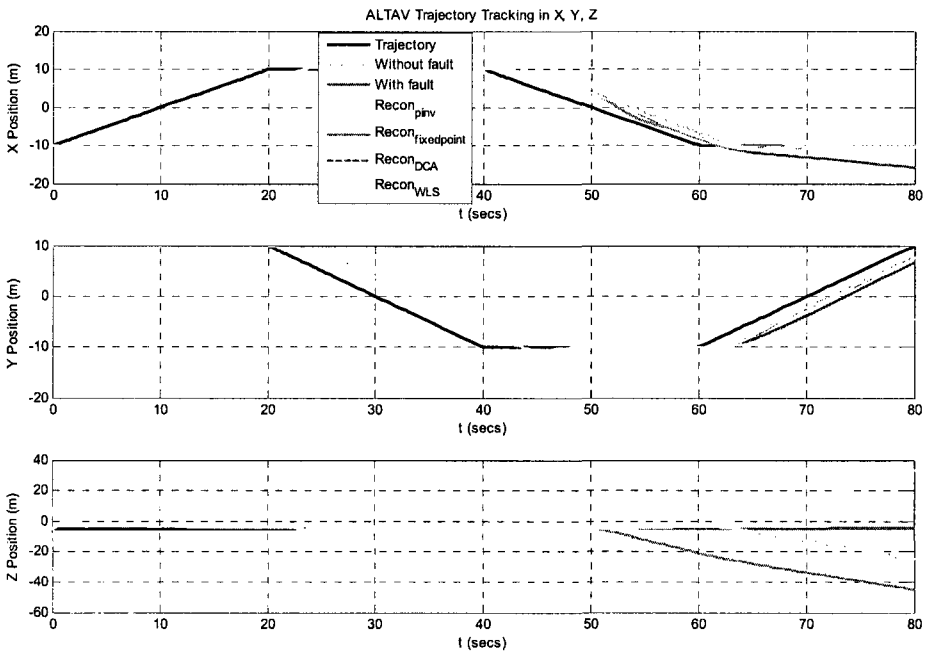

Figure 7. 7 Output response of X, Y, Z position

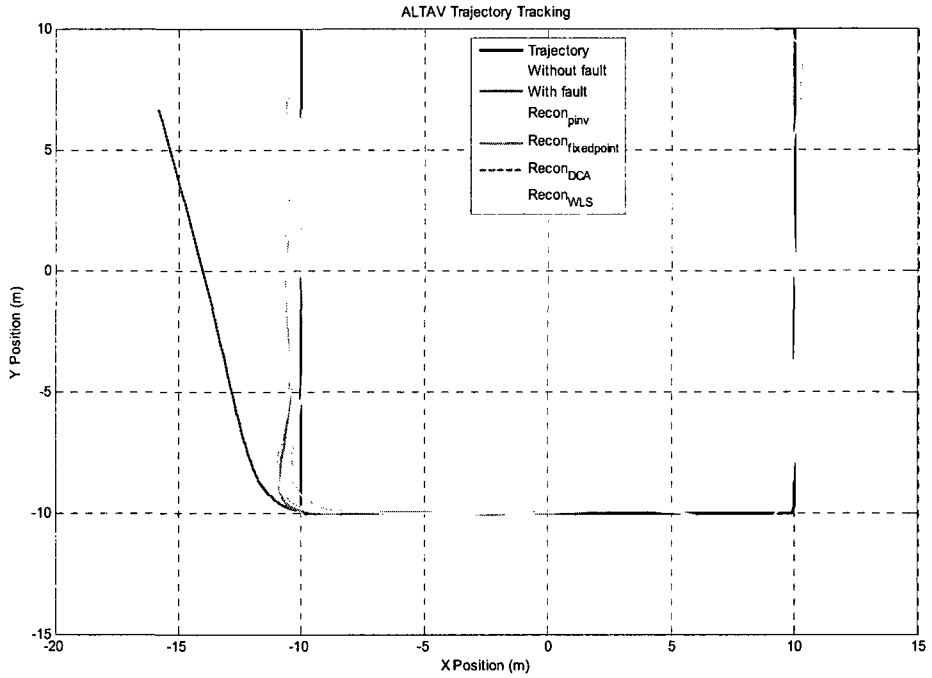

Figure 7. 8 UAV virtual and reallocation tracking trajectories

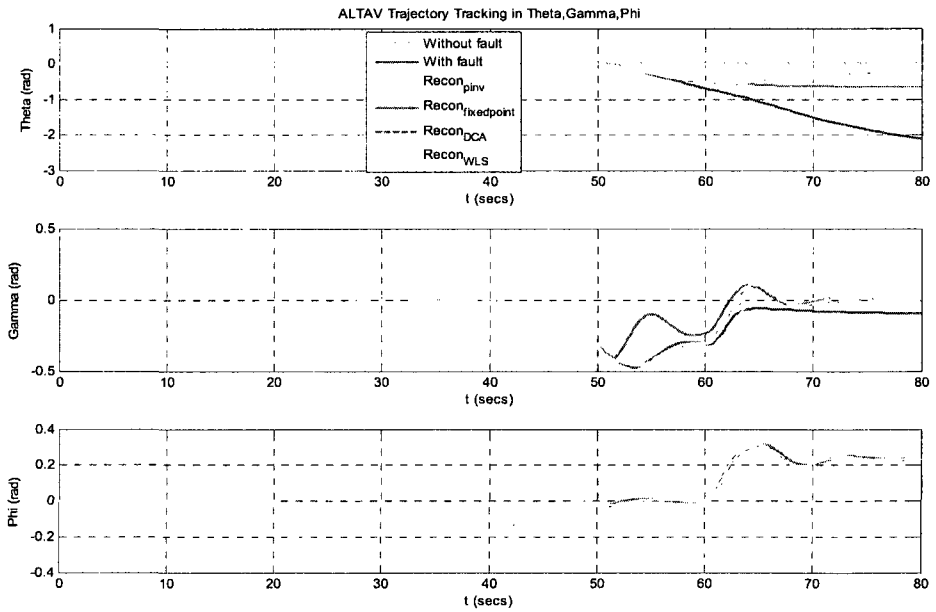

Figure 7. 9 Output responses of Theta, Gamma and Phi

As it can be seen from the above figures, the reconfigurable curves of fixed-point, DCA and WLS track the desired trajectory curve with a stable X position error. The configurable curves of CGI just track to the half of the fourth phase of the trajectory curves. In Figure 7.7, the Y position of the virtual trajectory curve using the CGI method should change from -10 to 10 when the model operate from 60s to 80s whereas the Y position of actual curves in the CGI method change from -10 to -3.5 when the model operates from 60s to 80s. On the other hand, the X position curves track the original trajectory without error, but with delay.

Scenario 2:

Input: Square input

Output: motor 3 with 50% loss

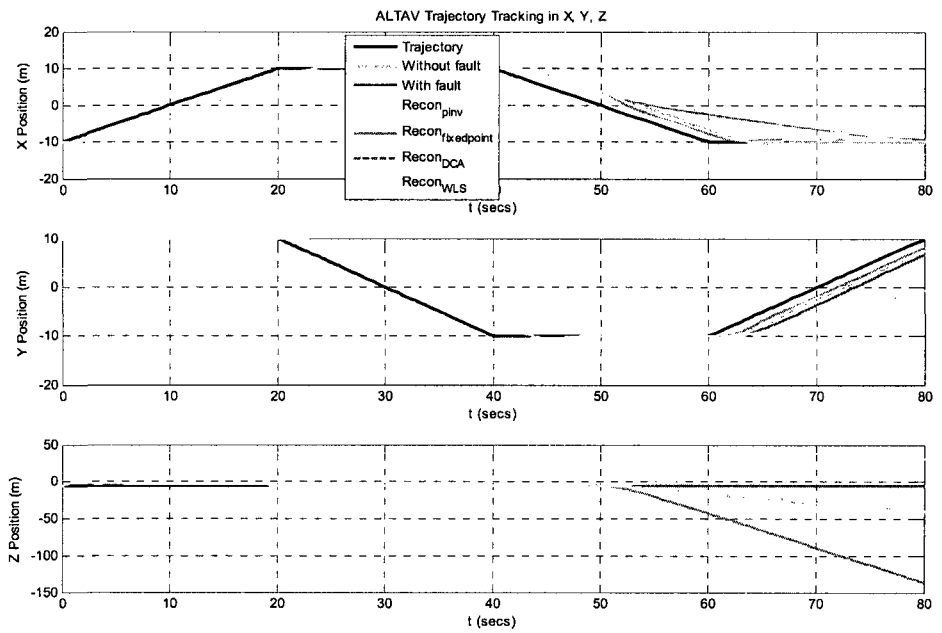

Figure 7. 10 Output response of X, Y, Z position

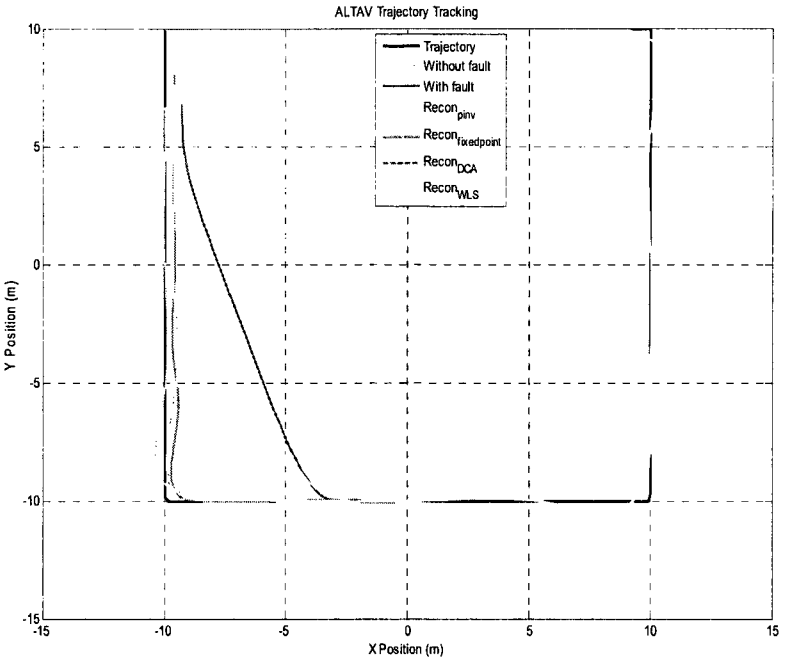

Figure 7. 11 UAV virtual and reallocation tracking trajectory

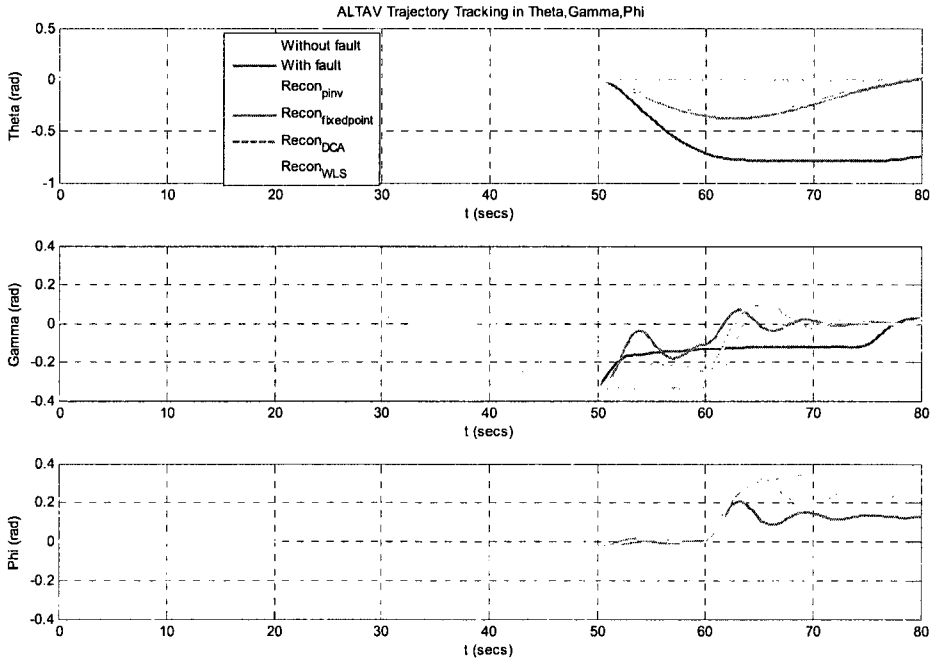

Figure 7. 12 Output responses of Theta, Gamma and Phi

In Figure 7.11, the reconfigurable curves of fixed-point, DCA and WLS track the desired trajectory curve with a stable X position error. The configurable curves of the CGI method just track to the half of the fourth phase of the trajectory curves. In Figure 7.10, X and Y position curves in Fix, DCA and WLS follow the trajectory without error, but with delay. The X position curve in CGI fails to follow as reconfiguration starts, but curves return to the desired trajectory as curves end. The Y position curve in the CGI method fails to follow the desired trajectory curves. The Y position of the virtual trajectory curve in the CGI method should change from -10 to 10 when the model operates from 60s to 80s, whereas the Y position of the actual curves using the CGI method change from -10 to 0 when the model operates from 60s to 80s.

Scenario 3:

Input: Square input

Output: motor 4 with 80% loss

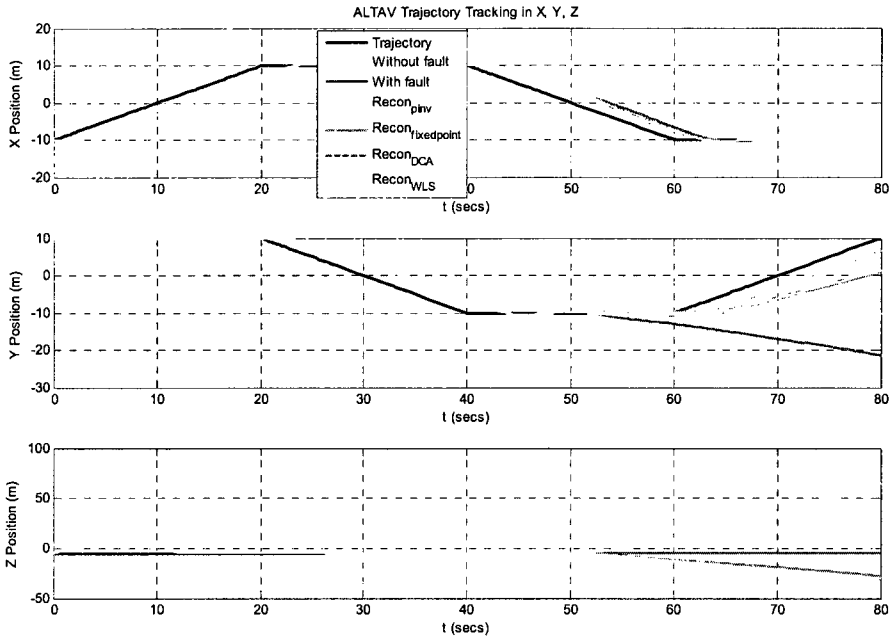

Figure 7. 13 Output response of X, Y, Z position

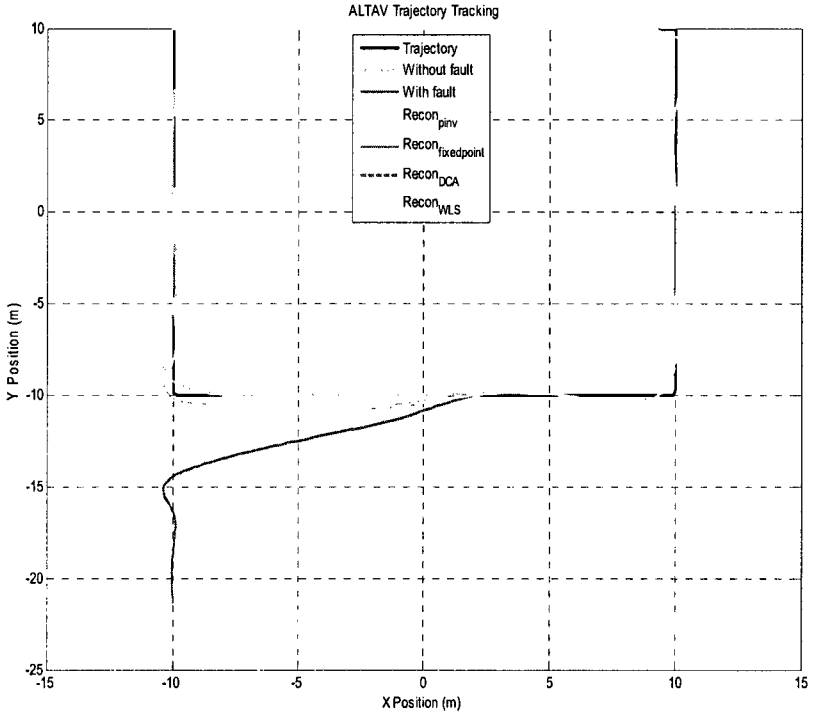

Figure 7. 14 Virtual and reallocation tracking trajectory

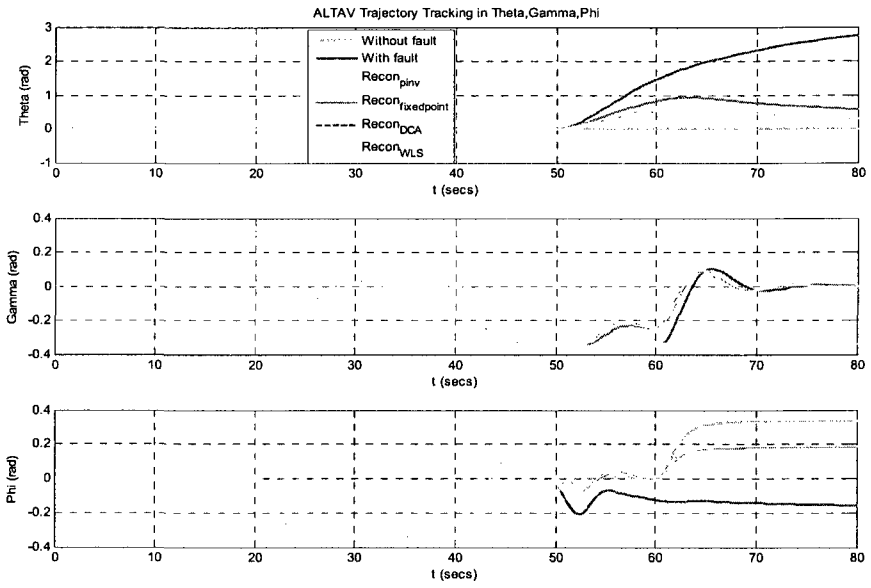

Figure 7. 15 Output responses of Theta, Gamma and Phi

In Figure 7.14, the reconfigurable curves of the fixed-point, DCA and WLS operate on the first half of the fourth phase of trajectory, whereas the reconfigurable curves of the CGI operate on the first half of the third phase of trajectory. In Figure 7.13, the X position reconfigurable curves of fixed-point, DCA and WLS track the trajectory curves with a delay, whereas the X position curves in the CGI follow the trajectory curve with a steady position error. The Y position reconfigurable curves in all methods have a position error when the model operates from 60s to 80s.

7.3.2. Circle trajectory as model input

Scenario 1:

Input: Circle input

Output: motor 2 with 80% loss

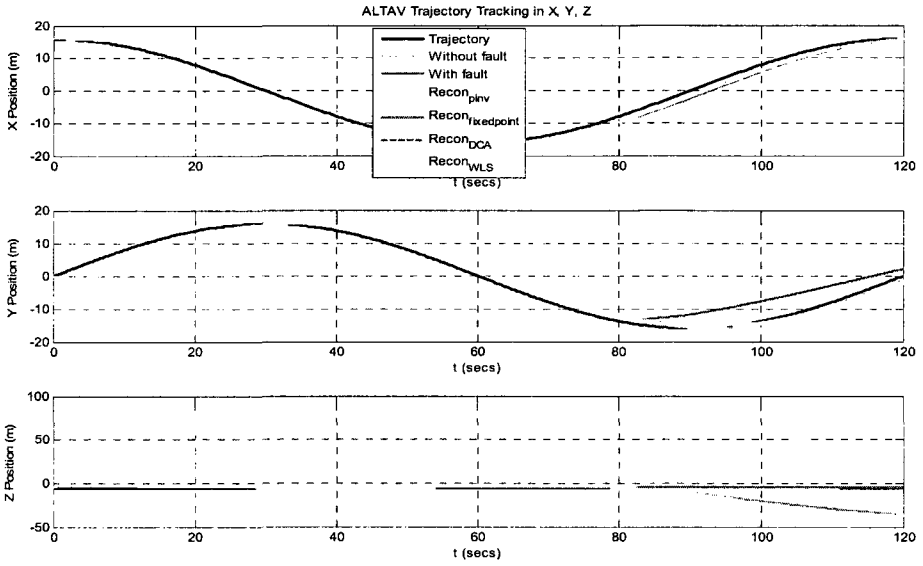

Figure 7. 16 Output response of X, Y, Z position

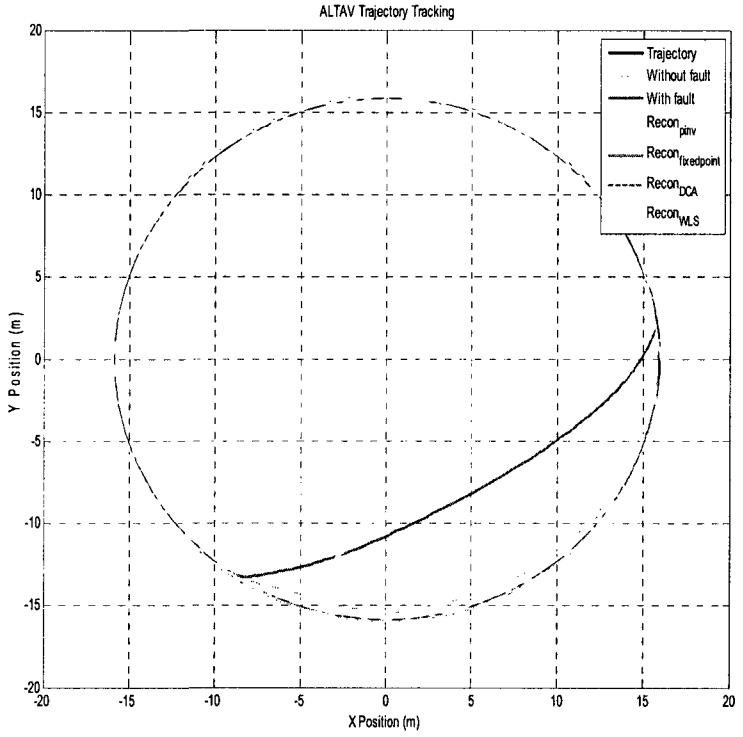

Figure 7. 17 UAV virtual and reallocation tracking trajectory

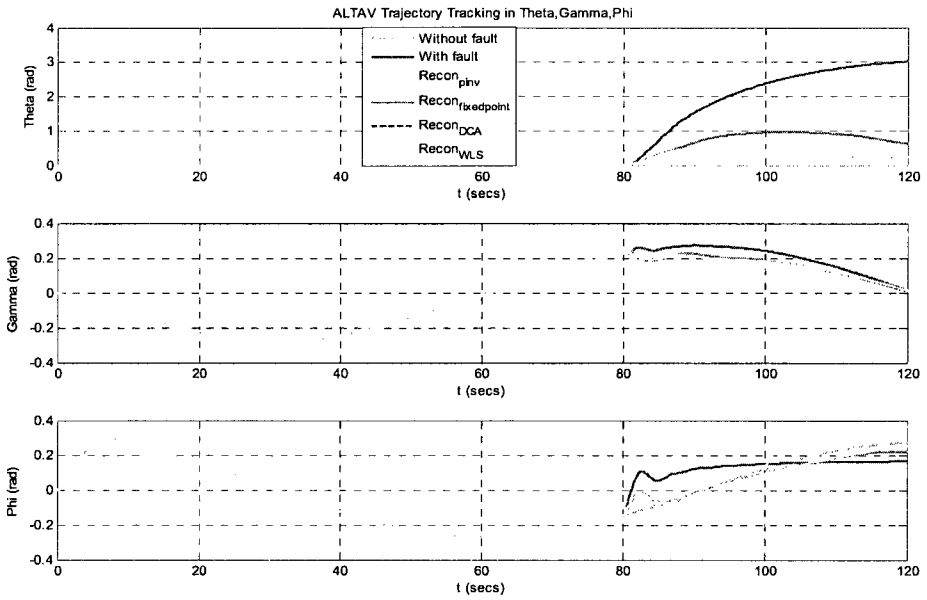

Figure 7. 18 Output responses of Theta, Gamma and Phi

In this scenario, the reconfigurable curves fixed-point, DCA and WLS follow the desired trajectory curves with a small error in Figure 7.17. The reconfigurable curves CGI fail to follow the trajectory curves.

Scenario 2:

Input: Circle input

Output: motor 3 with 90% loss

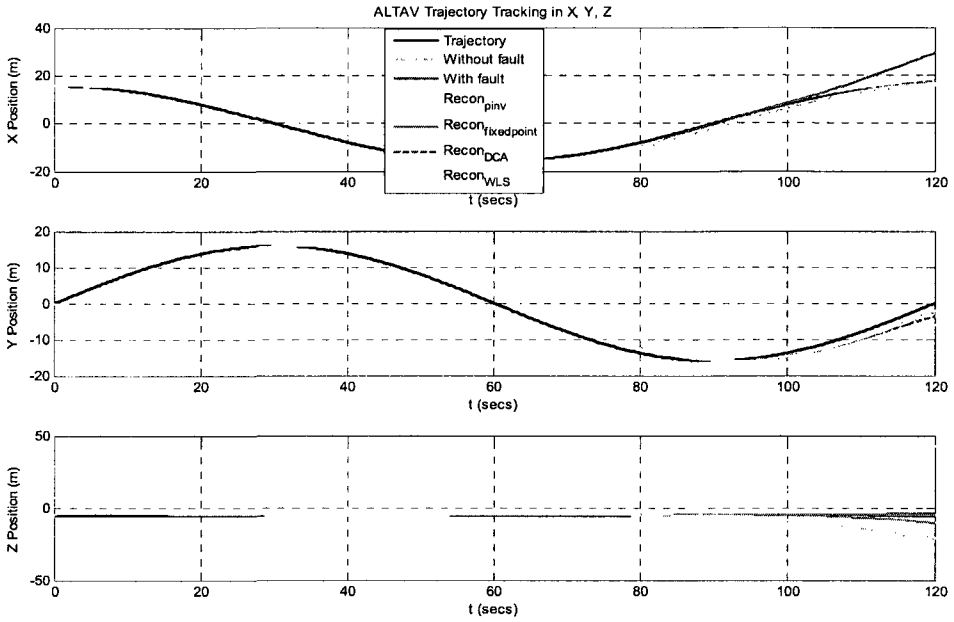

Figure 7. 19 Output response of X, Y, Z position

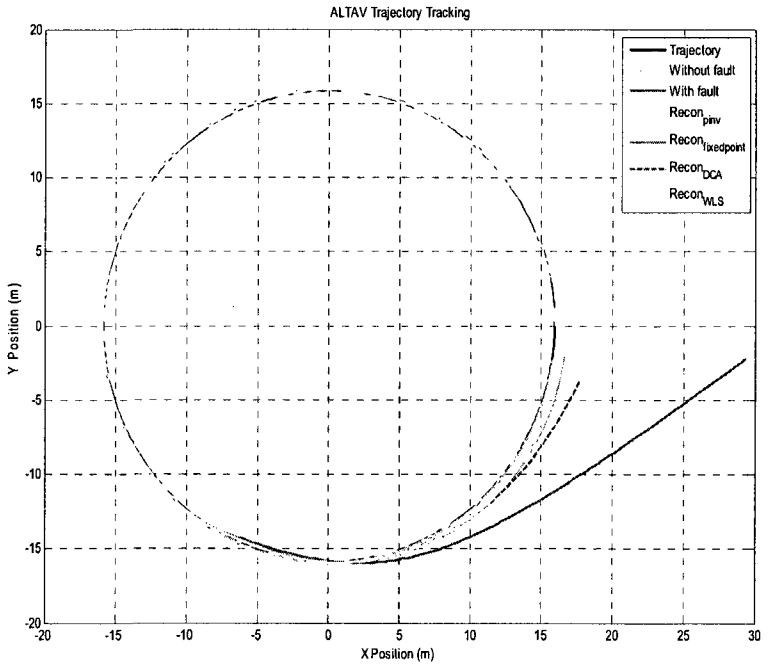

Figure 7. 20 UAV virtual and reallocation tracking trajectory

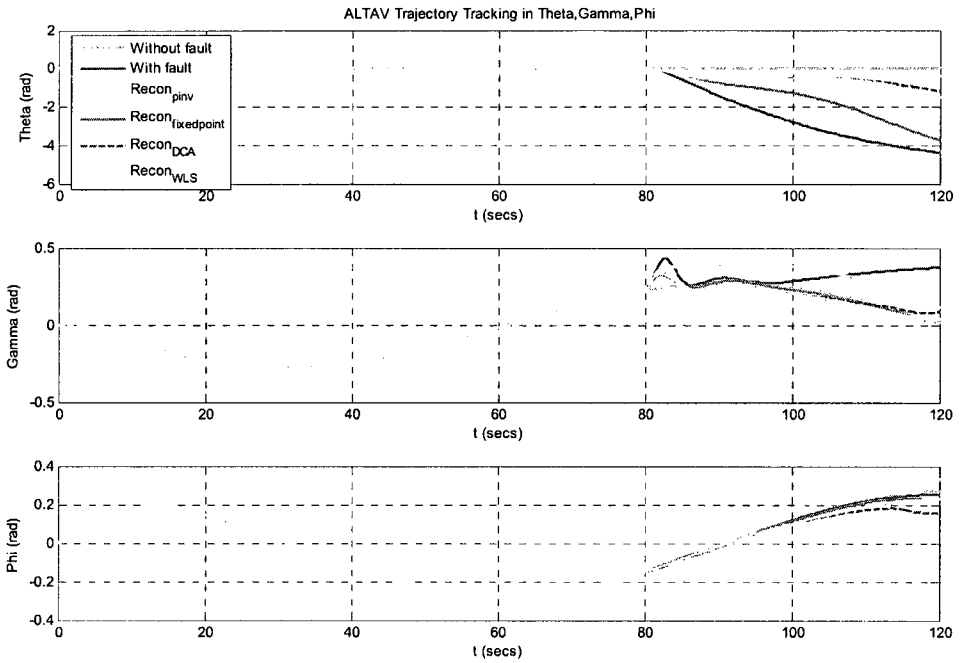

Figure 7. 21 Output responses of Theta, Gamma and Phi

As it can be seen from the above figures, the actual trajectory curves of the four different methods track the virtual trajectory with a small error, magnitude of error increase when reconfigurable time increases.

Scenario 3:

Input: Circle input

Output: motor 4 with 75% loss

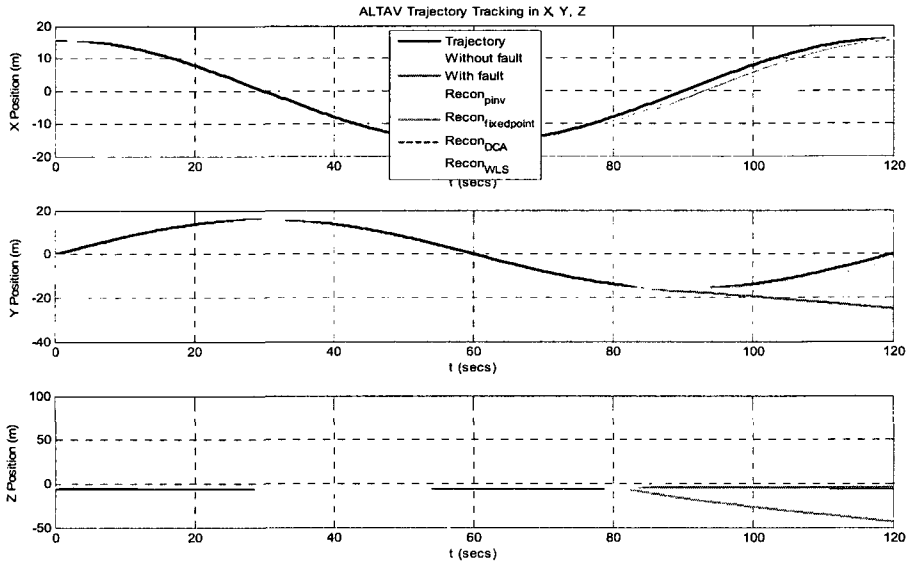

Figure 7. 22 Output response of X, Y, Z position

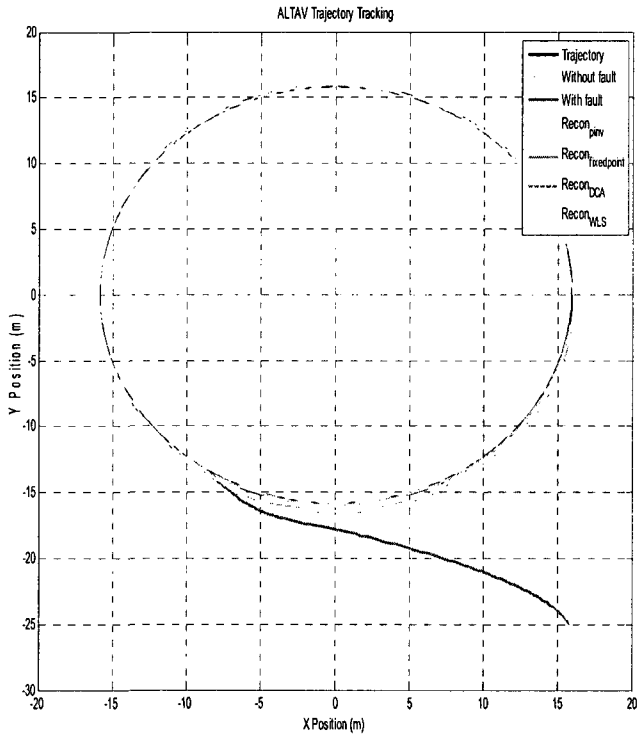

Figure 7. 23 UAV virtual and reallocation tracking trajectory

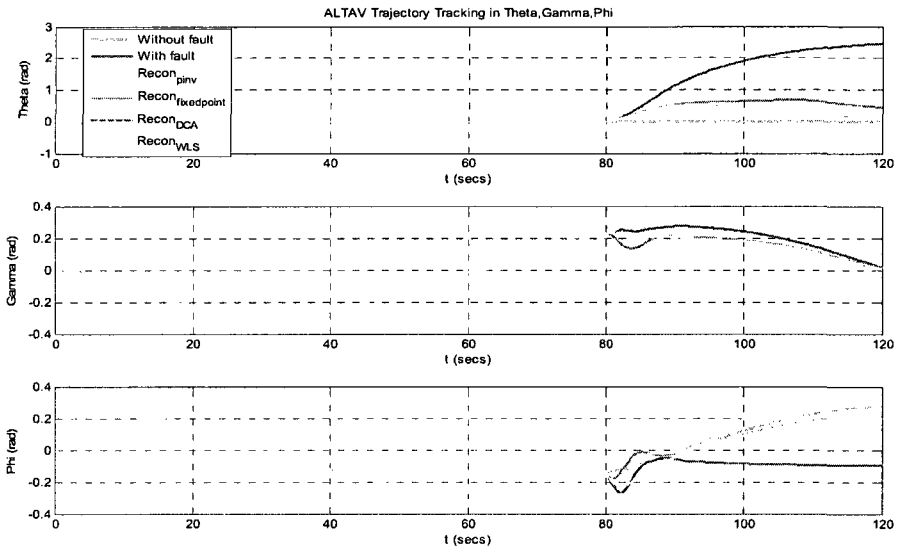

Figure 7. 24 Output responses of Theta, Gamma and Phi

In Figure 7.23, the reconfigurable curves of fixed-point, DCA and WLS track the trajectory curves with a small error and delay. The configurable curve of CGI fails to follow the virtual trajectory curve. In Figure 7.22, the X and Y position curves of fixed-point, DCA and WLS track the desired curves, whereas the X and Y position curves of CGI increases their magnitude error when reconfigurable time increases.

Summary for partial loss

The result shows that, when partial loss occurs in UAV motor, the performance is worse than the normal condition, and sometimes the fault is very critical. But the control reallocation technique gives a better solution for the loss and the UAV can track the normal situation without affecting performance. On the whole, the fixed-point, DCA and WLS methods show a better result and help the UAV to recover

from the fault and fly more safety than the CGI method regardless of square or circle input.

7.4. Simulation Results for Stuck Faults

In this section, the simulation results for stuck faults are shown for the UAV model without Gaussian noise.

Stuck fault scenario is considered as follows:

Input command: Square trajectory as model input

Scenario	Stuck Fault
1	Motor 1 stuck at an unknown position
2	Motor 3 stuck at the current position

Input command: Circle trajectory as model input

Scenario	Stuck Fault
1	Motor 2 stuck at an unknown position
2	Motor 4 stuck at the current position

7.4.1. Square trajectory as model input

Scenario 1:

Input: Square input

Fault: Motor 1 stuck at an unknown position

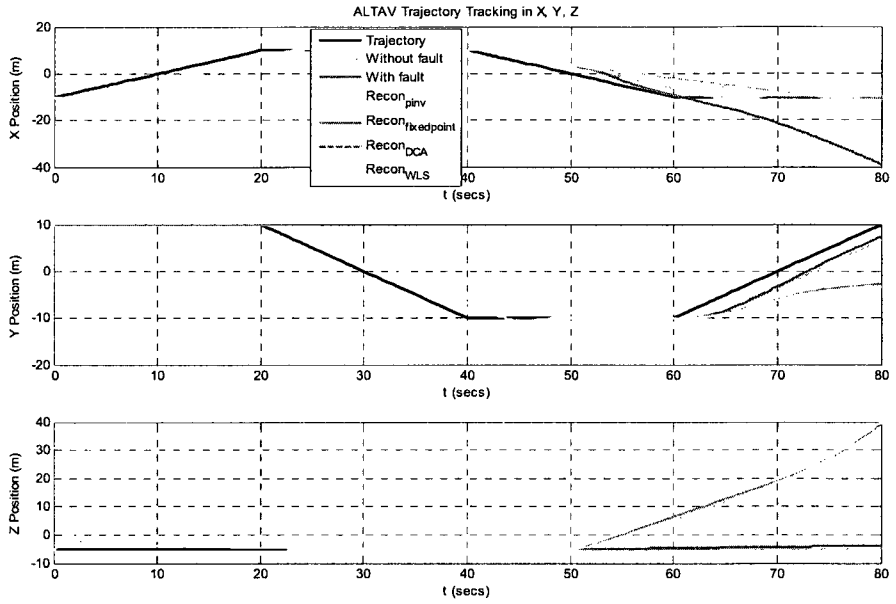

Figure 7. 25 Output response of X, Y, Z position

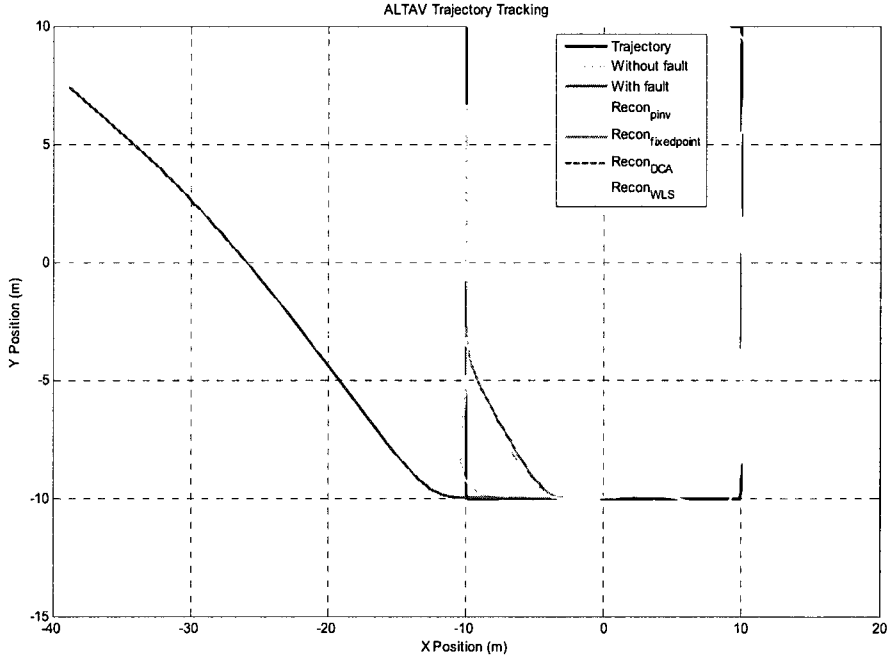

Figure 7. 26 UAV virtual and reallocation tracking trajectory

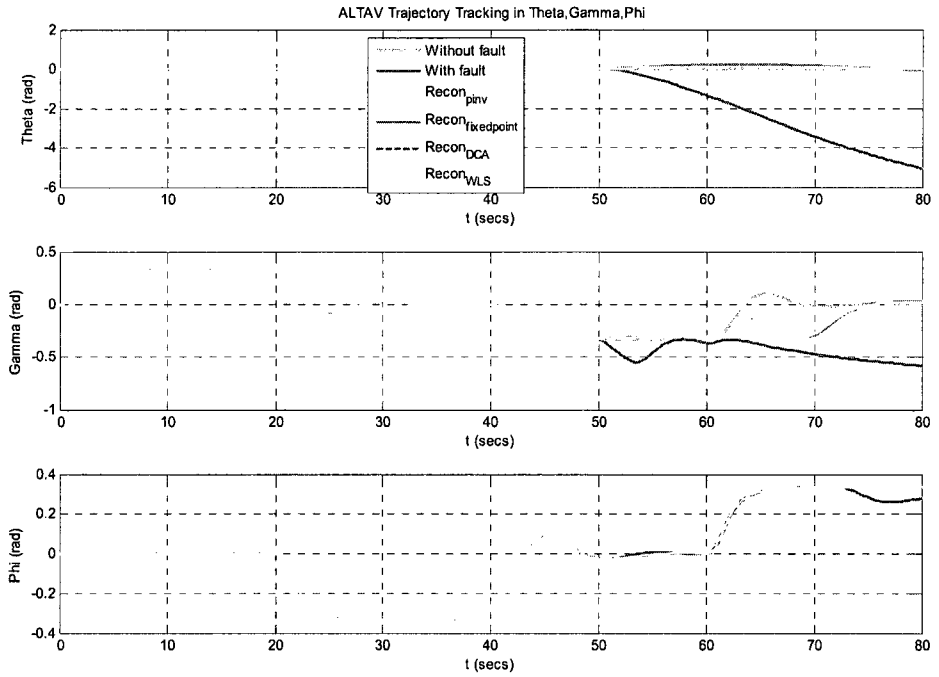

Figure 7. 27 Output responses of Theta, Gamma and Phi

In this scenario, all four control reallocation methods do not ideally track the virtual trajectory and the actual trajectories operate on the first half of the fourth phase of trajectory. The reason is that from Figure 7.25, the Y position curves fail to follow the desired trajectory curves. The Y position of the virtual trajectory curve should change from -10 to 10 when the model operates from 60s to 80s, whereas the Y position of the actual curves of control reallocation methods change from -10 to -3 when the model operate from 60s to 80s.

Scenario 2:

Input: Square input

Fault: Motor 3 stuck at the current position

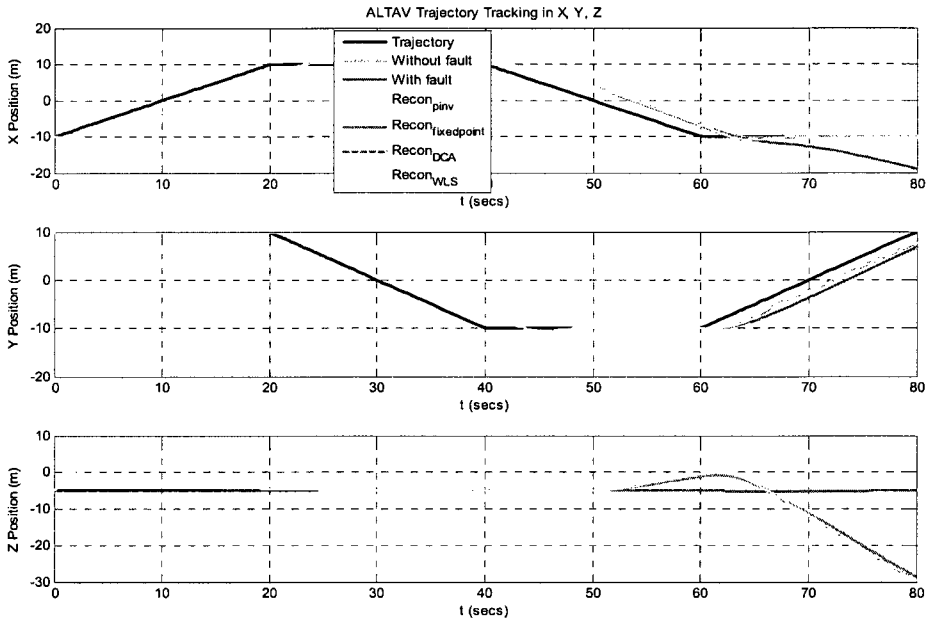

Figure 7. 28 Output response of X, Y, Z position

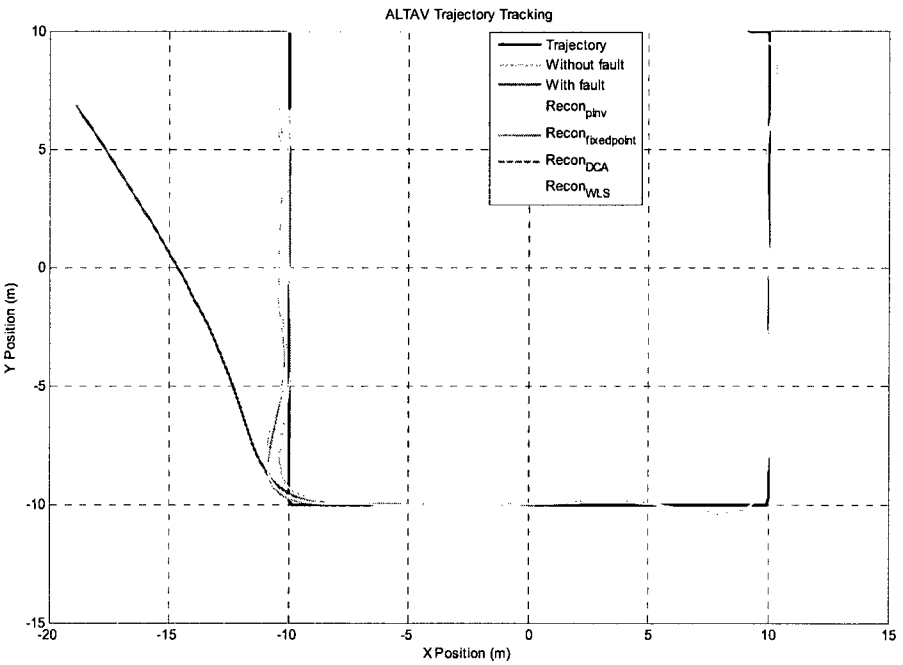

Figure 7. 29 UAV virtual and reallocation tracking trajectory

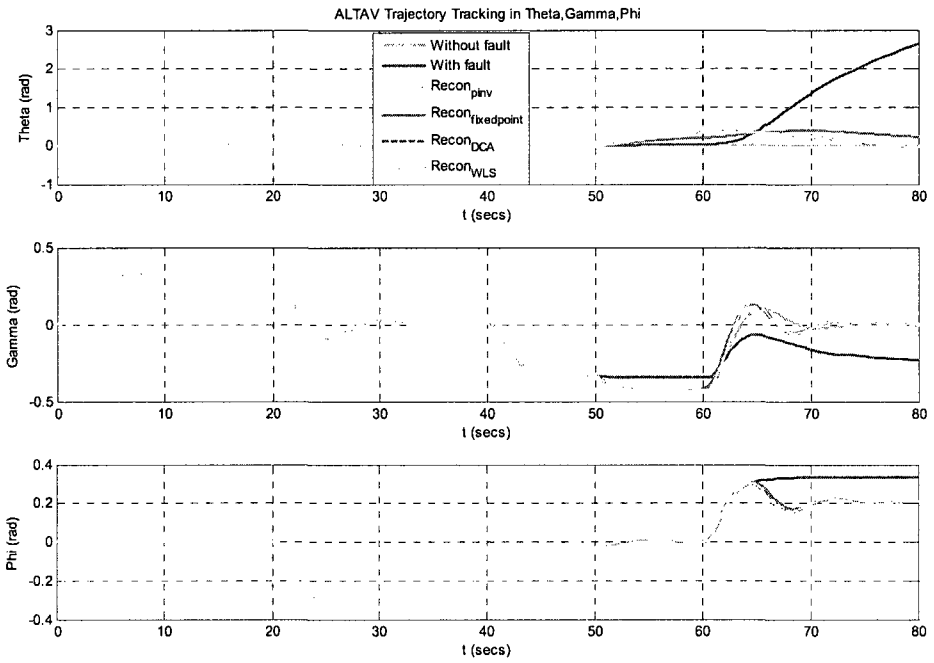

Figure 7. 30 Output responses of Theta, Gamma and Phi

It can be seen that the control reallocation curves of four different methods follow the desired trajectory curves with a delay and small error.

7.4.2. Circle trajectory as model input

Scenario 1:

Input: Circle input

Fault: Motor 2 stuck at an unknown position

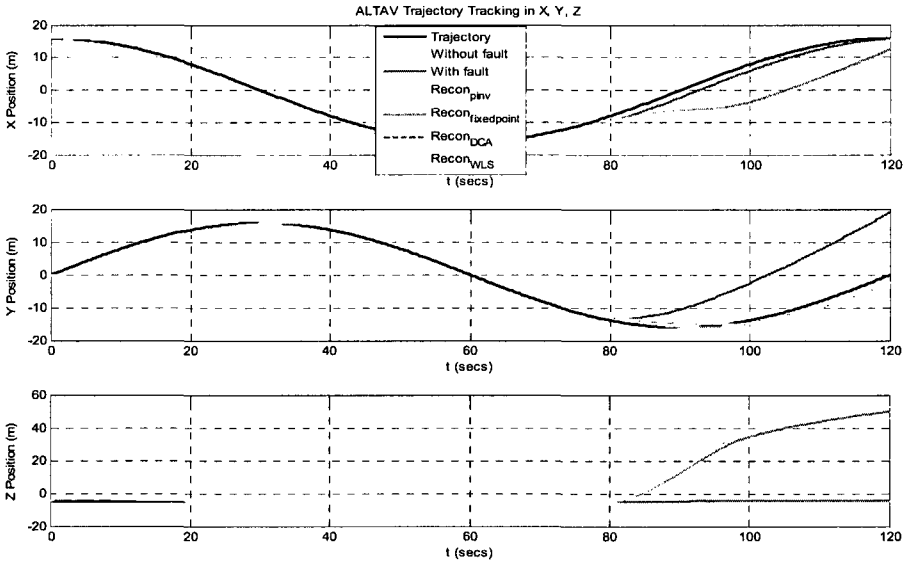

Figure 7. 31 Output response of X, Y, Z position

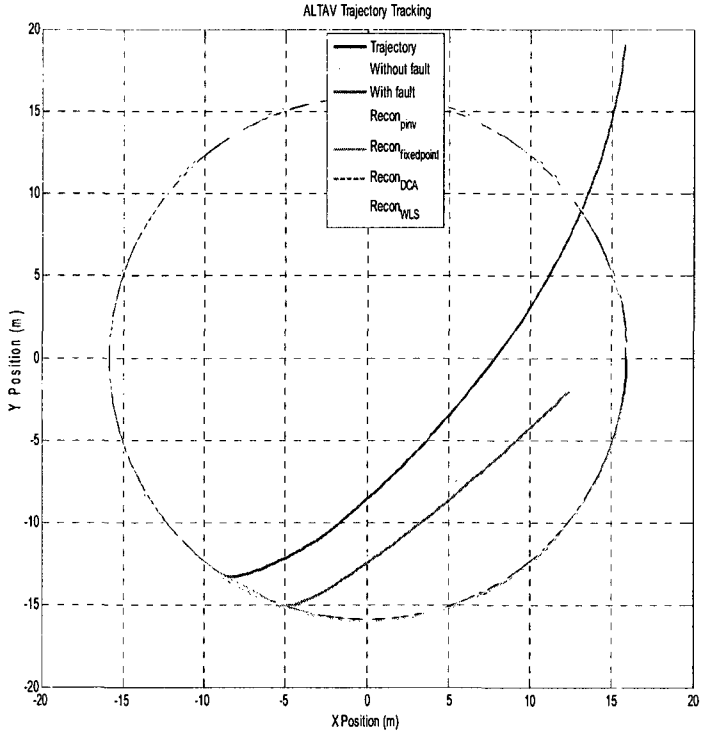

Figure 7. 32 UAV virtual and reallocation tracking trajectory

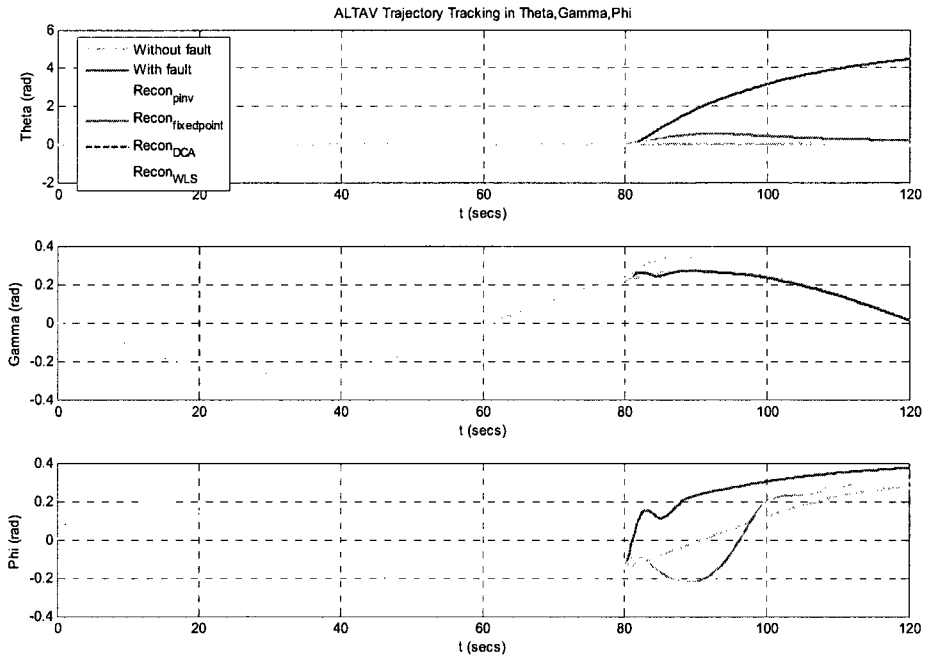

Figure 7. 33 Output responses of Theta, Gamma and Phi

In this scenario, the reconfigurable curves of the four methods fail to follow the desired trajectory curves, just compensate some fault.

Scenario 2:

Input: Circle input

Fault: Motor 4 stuck at the current position

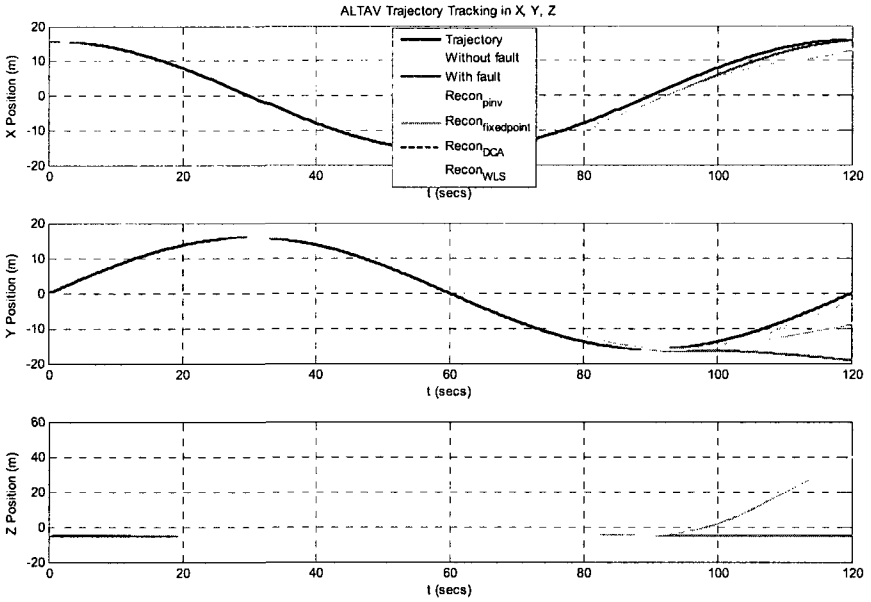

Figure 7. 34 Output response of X, Y, Z position

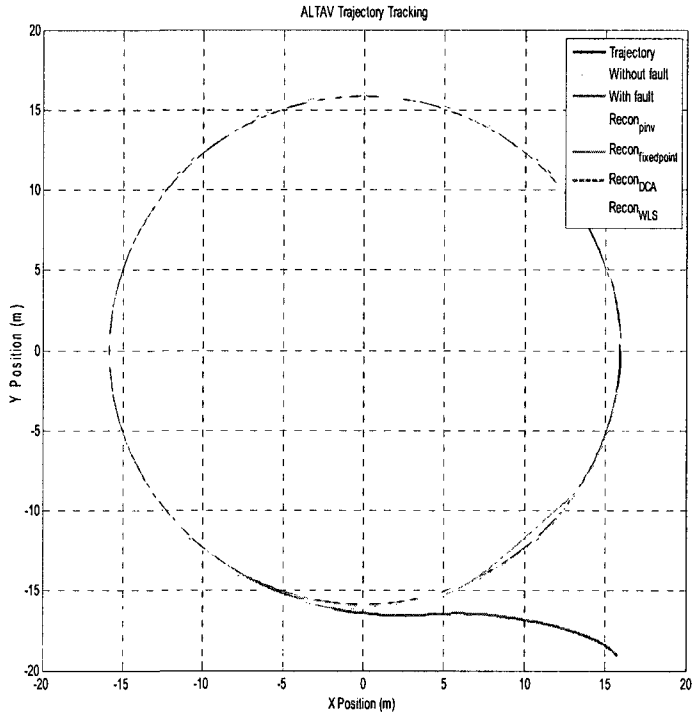

Figure 7. 35 UAV virtual and reallocation tracking trajectory

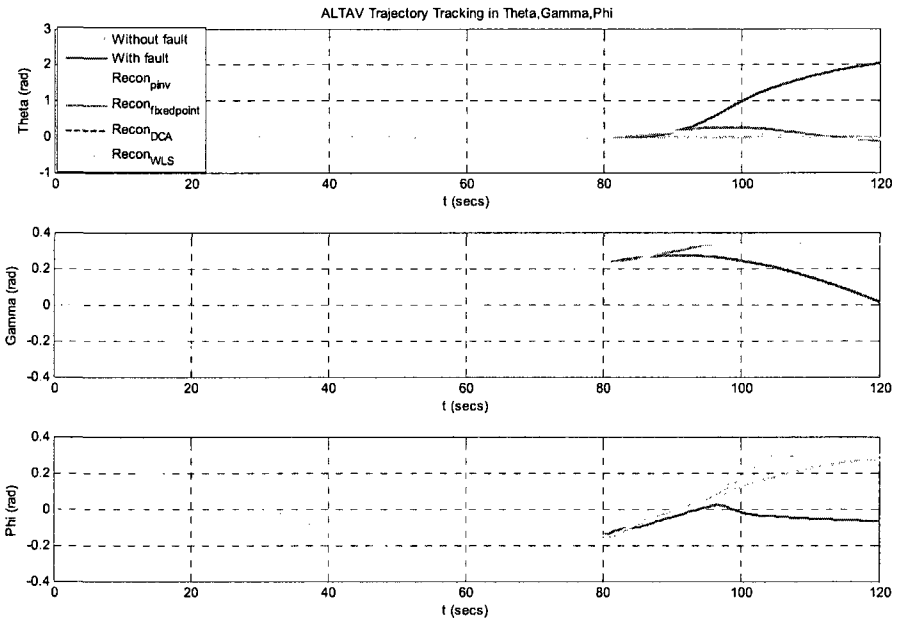

Figure 7. 36 Output responses of Theta, Gamma and Phi

From Figure 7.35, the four reallocation curves follow the desired trajectory curves with a small error and delay.

Summary for stuck fault

As can be seen from the results obtained, on the one hand, the control reconfigurable trajectories of four different algorithms follow the desired trajectories well, just with a delay and small error when system faults cause by stuck at the current position. The reason is that the X, Y reconfigurable position curves follow the desired position curves. On the other hand, all four control reconfigurable curves do not ideally track the desired trajectory, but improve the UAV tracking ability when system faults cause by stuck at an unknown position.

8. Simulation Results in ADMIRE

In this section, simulation results in different control reconfigurable algorithms are shown in the ADMIRE non-linear model. Their control input is different: the ALTAV model control input is given by a square or circle trajectory. In the ALTAV model, trajectory is selected by the operator or program. The ADMIRE model control inputs are given by the pilot, among them are longitudinal (Fes) and lateral (Fas) stick deflection, and rudder deflection (Frp). Step input is considered in this project.

For the ADMIRE aircraft model, simulation time is 10s, the fault is generated at 2s and control reconfiguration starts at 2s. Same as in the ALTAV model, the simulation results of the ADMIRE model are shown below. Different algorithms generate the signals for deflection on the aircraft's seven control surfaces.

Partial loss control effectiveness scenario is considered as follows:

Scenario	Control Input	Partial Loss
1	Longitudinal stick, step 50N	Right canard 85% loss
2	Lateral stick, step -30N	Left outer elevon 40% loss

Stuck fault scenario is considered as follows:

Scenario	Control Input	Stuck Fault
1	Longitudinal stick, step 50N	Right canard stuck at 1 deg
2	Lateral stick, step -30N	Left outer elevon stuck at 1 deg

Same as for ALTAV, for easy comparison with all the different control re-allocation algorithms, together with normal and faulty responses the simulation result includes six curves in each graph. These curves are:

Green:	Without fault
Blue plus solid line:	Fault
Cyan:	Reconfiguration with CGI method
Magenta:	Reconfiguration with Fix method
Red plus dash line:	Reconfiguration with DCA method
Black plus dash line:	Reconfiguration with WLS method

8.1. Partial Loss Simulation Result

The following simulation results are related to partial loss for the ADMIRE non-linear aircraft model.

Scenario 1:

Input: Longitudinal stick deflection, step 50N as control input

Fault: Right canard with 85% loss

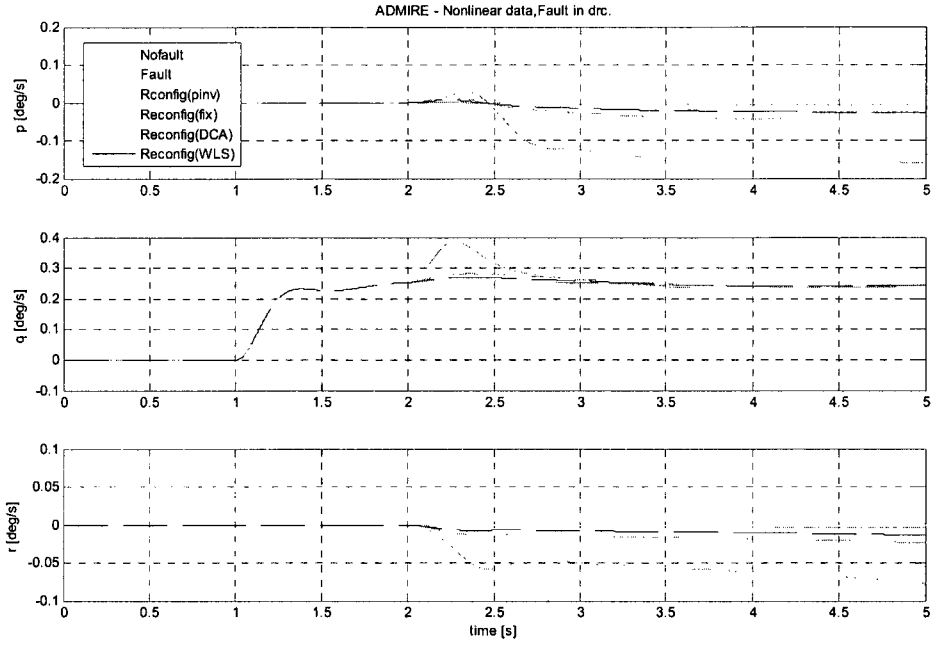

Figure 8. 1 Response of p, q, r

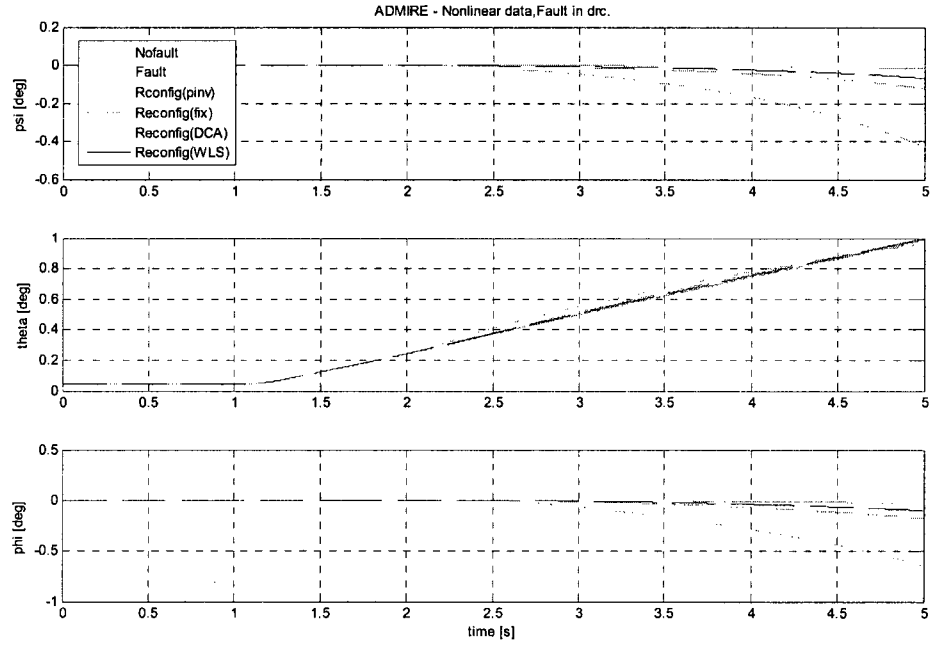

Figure 8. 2 Response of Euler angles ψ, θ, ϕ

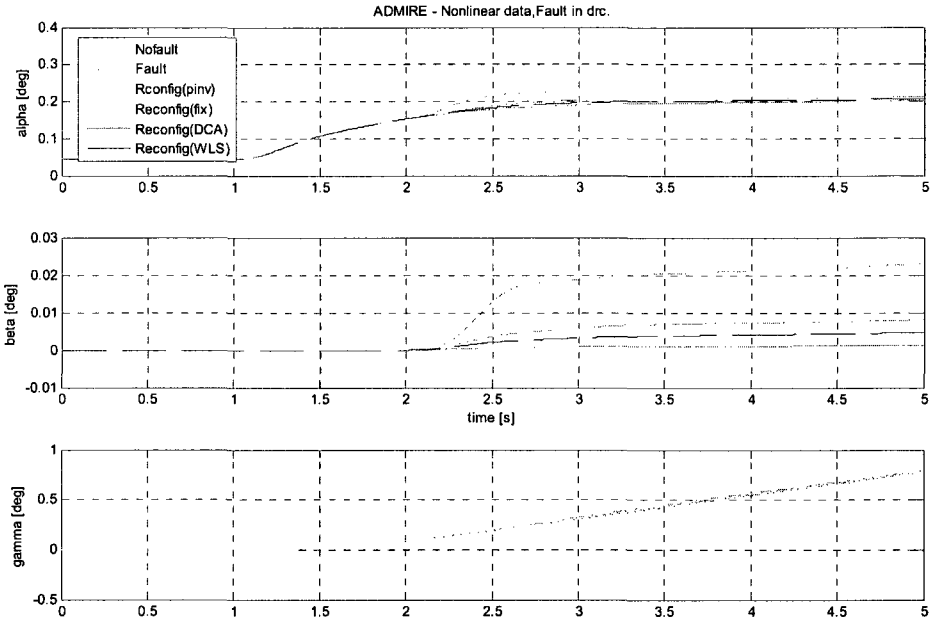

Figure 8. 3 Response of α, β, γ

In this scenario, the reconfigurable curves of the CGI, DCA and WLS methods track the desired output response with a small error, whereas the fix method follows the output response with a steady state error in Figure 8.1 and 8.3, with increasing magnitude in Euler angles until simulation runs in Figure 8.2.

Scenario 2:

Input: Lateral stick deflection, step -30N as control input

Fault: Left outer elevon with 40% loss

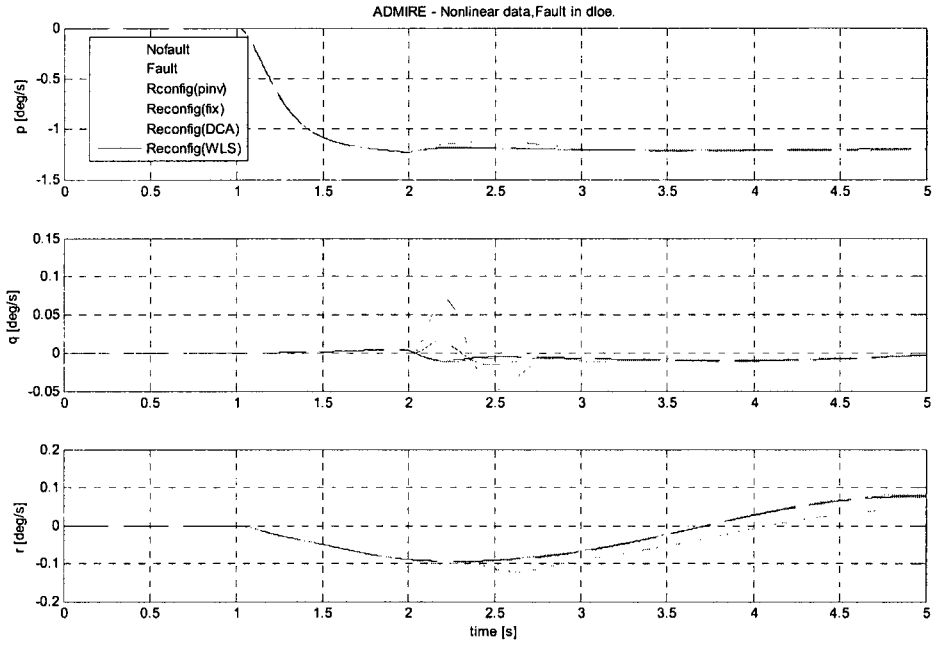

Figure 8. 4 Response of p, q, r

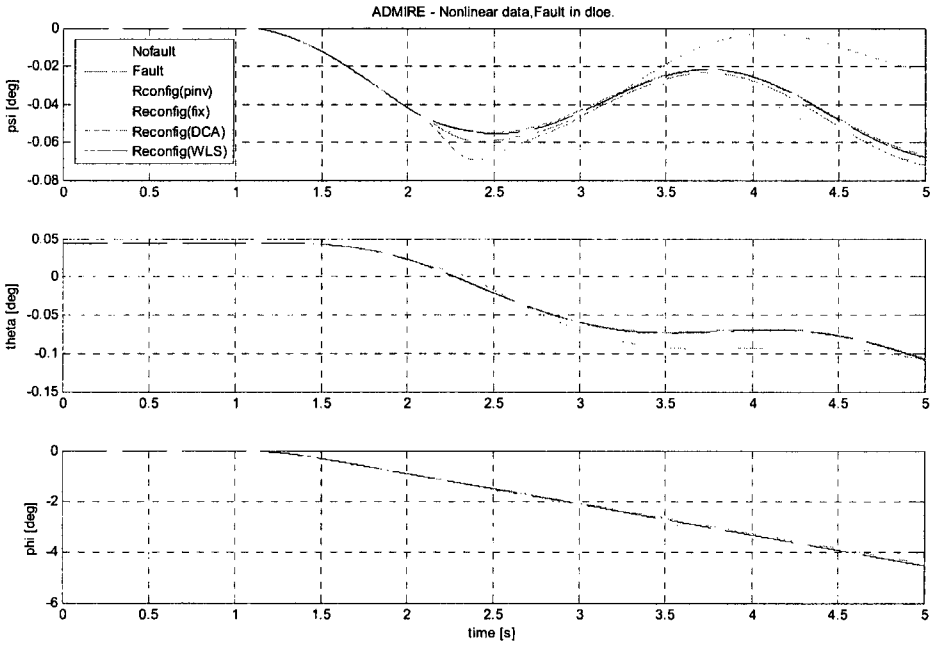

Figure 8. 5 Response of Euler angles ψ, θ, ϕ

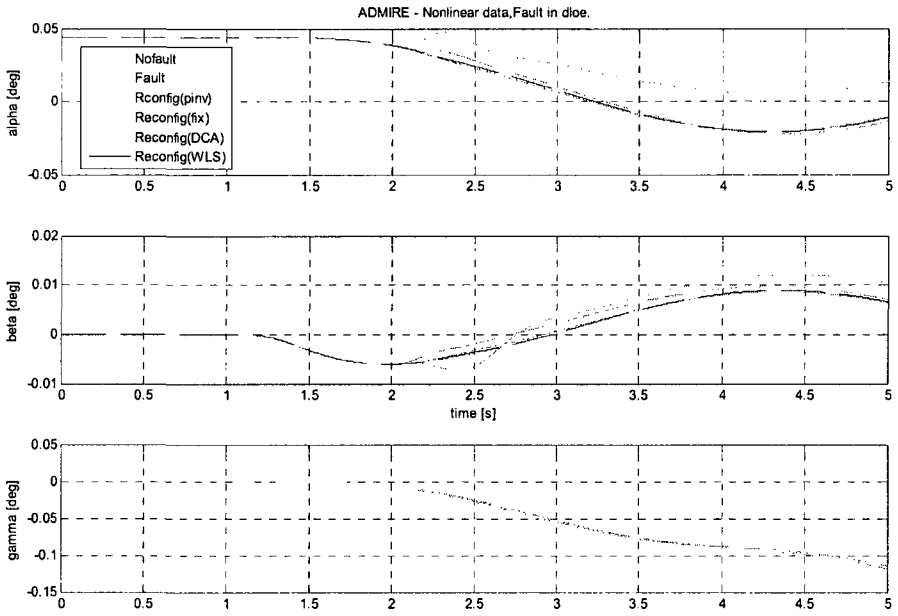

Figure 8. 6 Response of α, β, γ

From the above figures, it can be seen that the reconfigurable curves of the CGI, DCA and WLS methods track the desired output response with a small error. The Fix curve follows the desired output response with a steady-state error, whereas its magnitude increase in Euler angles ψ in Figure 8.5.

Summary for partial loss

It can be clearly seen from the results obtained, when partial loss occurs, the aircraft performance degrades from the normal condition. But the reconfigurable control technique gives a better solution for the loss and maintains the aircraft to follow the normal situation without degrading the performance. The CGI, DCA and WLS methods show better results and help the aircraft to recover from the fault and give a

safety flying than Fix method.

8.2. Simulation Results for stuck Faults

The following simulation results are related to stuck faults for ADMIRE non-linear aircraft model.

Scenario 1:

Input: Longitudinal stick deflection, step 50N as control input

Fault: Right canard stuck at 1 deg

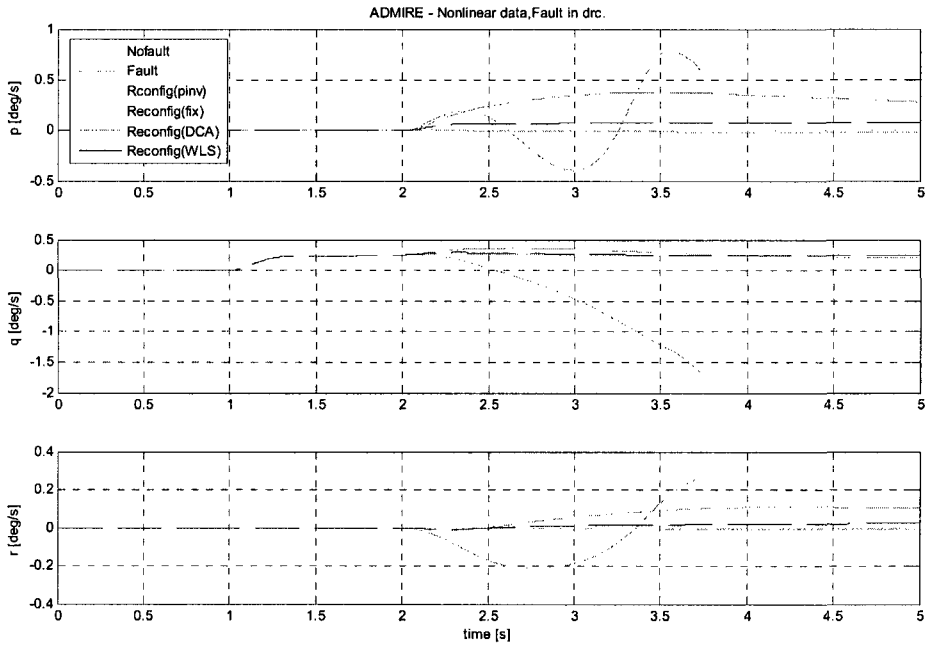

Figure 8. 7 Response of p, q, r

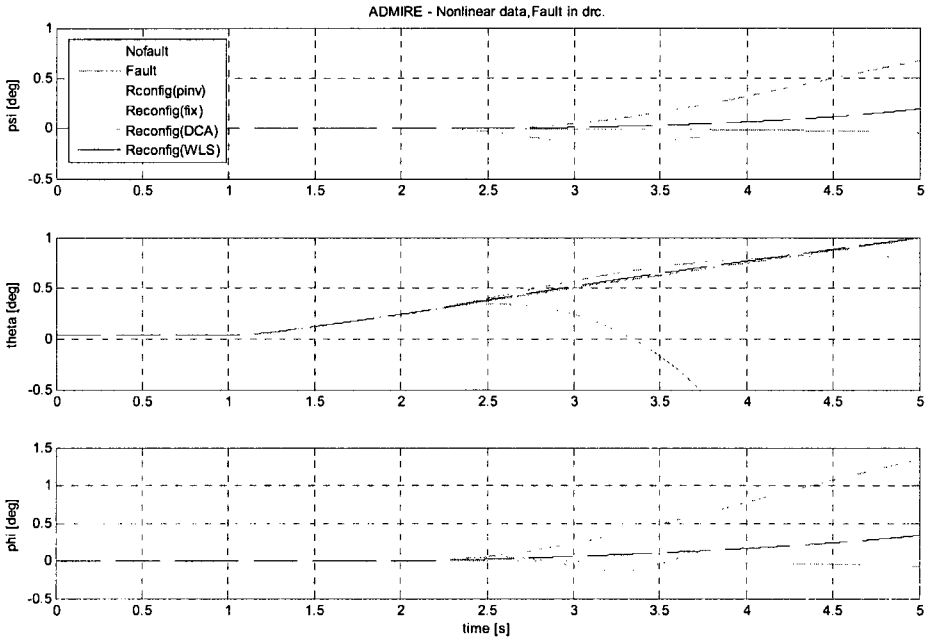

Figure 8. 8 Response of Euler angles ψ, θ, ϕ

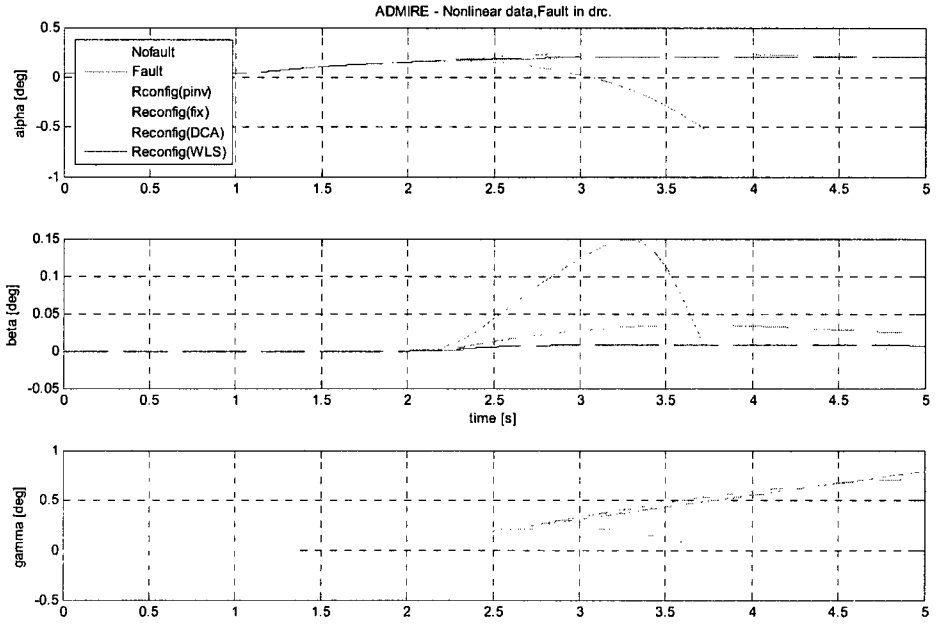

Figure 8. 9 Response of α, β, γ

From the above figures, the reconfigurable curves of CGI and WLS track the desired output response, the fix method can not complete the simulation and stops at 3.7s, the DCA method tracks the desired output response with a steady error in Figures 8.7 and 8.9, whereas its magnitude increases in Figure 8.9.

Scenario 2:

Input: Lateral stick deflection, step -30N as control input

Fault: Left outer elevon stuck at 0.1 deg

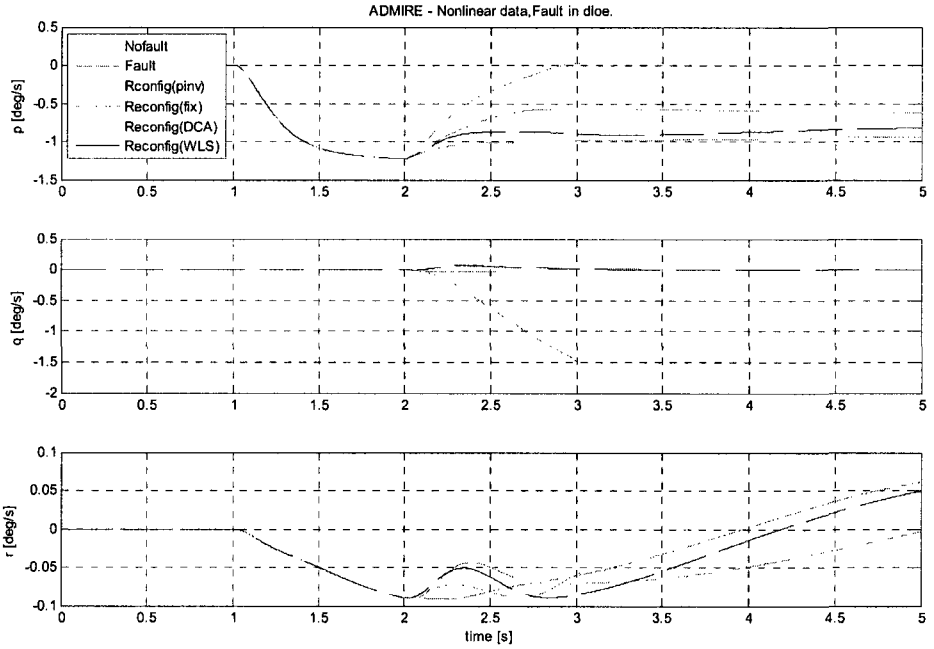

Figure 8. 10 Response of p, q, r

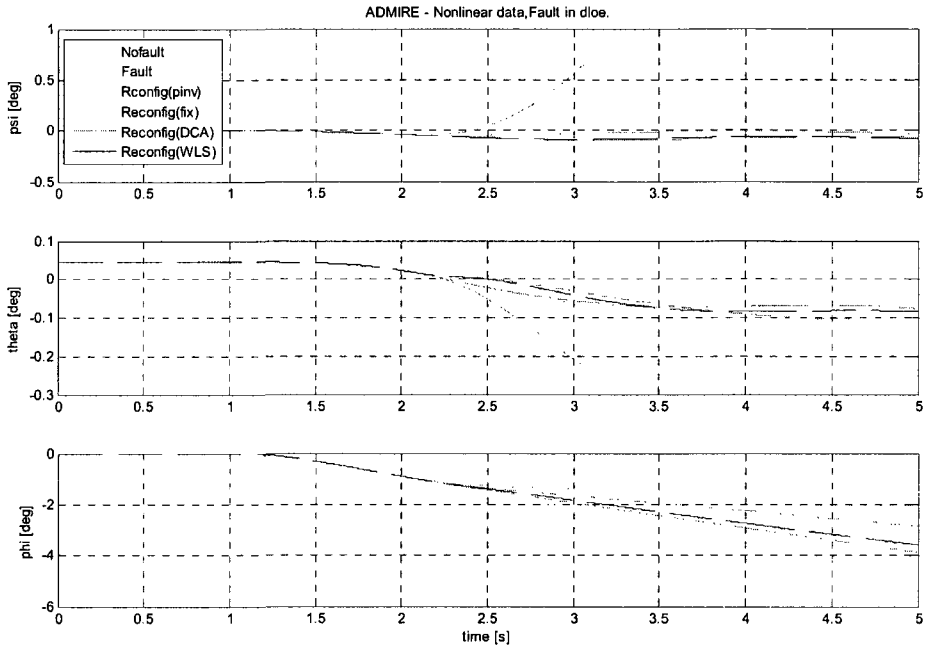

Figure 8. 11 Response of Euler angles ψ, θ, ϕ

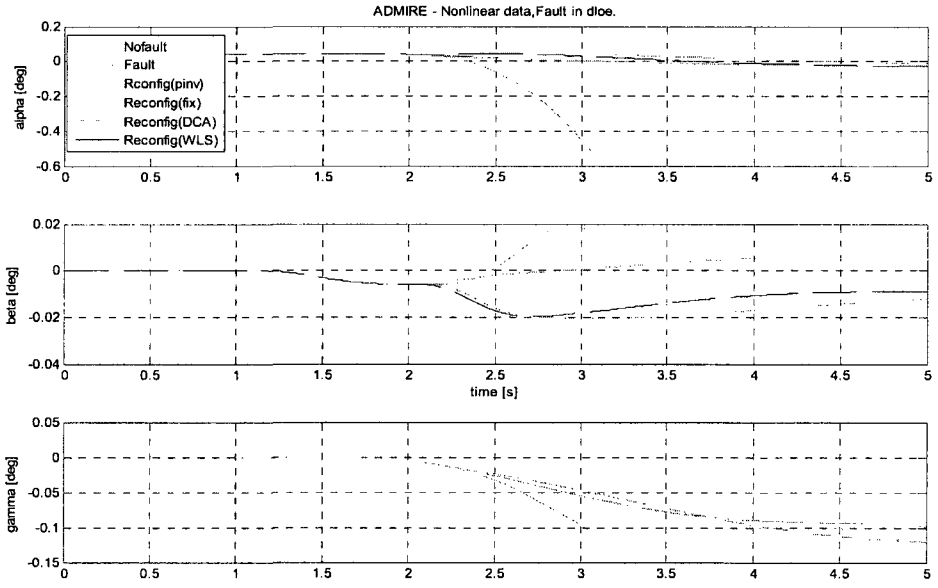

Figure 8. 12 Response of α, β, γ

In this scenario, the reconfigurable curves of CGI, WLS and DCA track the desired output response, whereas p in Figure 8.10 and β in Figure 8.12 have steady errors, fix method can not complete the simulation and stops at 3.1s.

Summary for stuck fault

The results shows, the output response from CGI, WLS and DCA methods track the desired output response of the aircraft, and even some variables have steady errors. The Fix algorithm exceeds the saturation limit and making the system unstable. This leads to sudden change in the control surface loosing the track of desired output response results in degrading the performance of the aircraft.

9. Conclusion and Future Work

Reconfigurable flight control research is an important development direction in future UAV flight control systems. It possesses a profound study value and wide application prospect. Currently, it has become the focus of the high-tech field of aviation research. This work, combined with the aircraft requirements of the pre-research project, analyzed the performance of the ALTAV and ADMIRE control reallocation technique under the presence of partial loss and stuck faults. Simulation and implementation of the test scenarios were conducted in the UAV ALTAV and ADMIRE benchmark models. The test scenarios of ALTAV were carried out in non-linear environments of the benchmark models with and without real world correction. There were two different flight paths as model trajectories in each environment of the benchmark. The test scenarios of ADMIRE were implemented in non-linear environment. The model input was the step input coming from longitudinal and lateral stick deflection. The results have been shown under the influence of partial loss and stuck at unknown positions in both the ALATV and ADMIRE environments.

A reconfigurable control allocation technique for flight control system to handle the fault was obtained. A detailed description about the control re-allocation technique has been presented. Implementation of fault scenarios in the ALTAV and ADMIRE benchmark models has been demonstrated. Four reconfigurable methods: cascaded generalized inverse, fixed-point, direct control allocation and weighted least squares,

were used to distribute the control surfaces to provide the desired control moments and forces for the fault UAV and aircraft.

For the UAV ALTAV model, following the results presented the overall performance of Fix, DCA and WLS show an effective tracking of the command input of the UAV, leading to minimum error compared to the CGI method for partial loss in the control motor. For stuck fault, the algorithms have better performance if system faults cause by stuck at a current position.

For the aircraft ADMIRE model, the reconfigurable methods of CGI, DCA and WLS track the desired output response better than the Fix method for partial loss on control surface. The simulation performance of the CGI and WLS methods are better than Fix and DCA in stuck faults.

In my work, my contribution is that an initial study of reconfigurable control allocation is applied to a realistic and nonlinear UAV and fixed-wing aircraft models. The different control reallocation: the cascaded generalized inverse algorithm, fixed-point algorithm, direct control allocation algorithm and weighted least squares algorithm are implemented and tested under ALTAV UAV and ADMIRE aircraft benchmarks, designing two different trajectories selection as UAV control input. Different control actuator faults caused by partial loss and by stuck at a current or an unknown position are implemented in these two benchmarks and are used for evaluating the control reallocation scheme. When partial loss and

stuck faults occur in UAV/aircraft control surfaces, the performance degrades from the normal condition even though the fault is not very critical. However, the reconfigurable control techniques give a better solution for these two fault scenarios and maintain UAV/aircraft's ability to track the normal situation without degrading performance. Simulation results have shown satisfactory results for accommodating the partial loss and stuck failures. Implementation of four reconfigurable control allocation algorithms improve UAV and aircraft flight control system's reliability and survivability, enabling them to complete pre-planned missions and to land safely even if the flight control system has faults caused by control actuators/effectors.

Future works include the investigation and testing on other control reallocation methods, the extension from one UAV to a UAV formation, and the investigation on reconfigurable baseline control techniques, in conjunction with control allocation techniques to achieve improved performance. In addition, the incorporation of fault detection and diagnosis schemes in ALTAV and ADMIRE environment is also to be conducted.

10.References

1) Escareno J., Salazar-Cruz S. and Lozano R., "Embedded control of a four-rotor UAV," *In proc. of the 2006 American Control Conference*, Minneapolis, Minnesota, USA, June 14-16 3936-3, 2006, pp 941.

2) Earon, E., "Almost-Light-Than-Air-Vehicle Fleet Simulation," *Technical Report*, V. 2.1, Quanser Inc., Toronto, Canada, 2005.

3) Zhang, Y. M., Jiang, J., "An Active Fault-Tolerant Control System against Partial Actuator Failures," *IEEE Proceedings- Control Theory and Applications*, Vol. 149, no. 1, 2002. pp. 95-104.

4) Meskin, N., Jiang, T., Sobhani, E., Khorasani, K. and Rabbath, C.A. "A Nonlinear Geometric Fault Detection and Isolation Approach for Almost-Light-Than-Air-Vehicles," *American Control Conference*, 2007.

5) Zhang, Y. M., Suresh, V. S., Jiang, B. and Theilliol, D., "Reconfigurable Control Allocation against Aircraft Control Effector Failures," *IEEE Multi-conference on Systems and Control,* Singapore, 2007.

6) Burken, John J., Lu, P., Wu, Z., and Bahm, C. "Two Reconfigurable Flight-Control Design Methods: Robust Servomechanism and Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 24, no. 3, 2001. pp 482-493.

7) Yang, K. C. H., Yuh, J. and Choi, S. K., "Fault-Tolerant System Design of an Autonomous Underwater Vehicle-ODIN: an Experimental Study," *International Journal of Systems Science*, 1999, Vol. 30, no. 9, pp. 1011-1019.

8) Omerdic, E., and Toal, D., "Control Allocation of Over-Actuated Thruster-

Propelled Underwater Vehicles," *Mobile and Marine Robotics Research Group*, ECE Department, University of Limerick, Ireland.

9) Bodson, M., "Evaluation of optimization methods for control allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 25, no. 4, 2002, pp 703-711.

10) Davidson, J. B., Lallman, F. J. and Bundick, W. T., "Integrated Reconfigurable Control Allocation," *In proc. of AIAA Guidance, Navigation, and Control Conference*, Montreal, Canada, August 2001, pp. 1-11.

11) Boskovic, J. D., Bergstrom, S. E., and Mehra, R. K., "Retrofit Reconfigurable Flight Control in the Presence of Control Effector Damage," *Proceedings of the 2005 American Control conference*, June 2005, pp. 2652-2657.

12) Ahmed-Zaid, F., Ioannou, P., Gousman, K., and Rooney, R., "Accommodation of Failure in the F-16 Aircraft Using Adaptive Control," *IEEE Control Systems Magazine*, 1991, pp. 73–78.

13) Maybeck, P. S., and Stevens, R. D., "Reconfigurable Flight Control via Multiple Model Adaptive Control Methods," *IEEE Transactions an on Aerospace and Electronic Systems*, 1991, Vol. 27, no. 3, pp. 470-479.

14) Chen, W., and Jiang, J., "Fault-Tolerant Control against Stuck Actuator Faults," *IEEE Pro.-Control Theory Appl.,* Vol. 152, no. 2, March 2005.

15) Zhang, Y. M., and Jiang, J., "Fault Tolerant Control System Design with Explicit Consideration of Performance Degradation," *IEEE Transactions an on Aerospace and Electronic Systems*, 2003, Vol. 39, no. 3, pp. 838-848.

16) Liao, F., Wang, J. L., and Yang, G. H., "Reliable Robust Flight Tracking Control: An LMI Approach," *IEEE Transactions on Control Systems Technology,* 2002,

Vol. 10, no. 1, pp. 76-89.

17) Yang, G. H., Zhang, S. Y., Lam, J., and Wang, J. L., "Reliable Control Using Redundant Controllers," *IEEE Transactions on Automatic Control*, 1998a, Vol. 43, no. 1), pp. 1588-1593.

18) Veillette, R. J., and Medanic, J. V. and Perkins, W. R., "Design of Reliable Control System," *IEEE Transactions on Aerospace and Electronic Systems*, 1992, Vol. 37, no. 3, pp. 290-304.

19) Forssell, L., Nilsson, U., "ADMIRE The Aero-Data Model in a Research Environment version 4.0, Model Description," *FOI- Swedish Defence Research Agency, Technical Report*, December 2005.

20) "Failure Scenarios and Assessment Criteria for Fault Detection, Isolation and Reconfiguration Techniques," *GARTEUR Action Group FM (AG16),* 2004.

21) Durham, W. C., "Constrained Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 17, no. 4, 1993. pp 717-725.

22) Bordignon, K. A., "Constrained Control Allocation for Systems with Redundant Control Effectors," *Ph.D. Thesis*. Virginia Polytechnical Institute and State University, 1996.

23) Zhang, Y. M. and Jiang, J., "Bibligraphical review on reconfigurable fault-tolerant control systems," *In proc. Of the 5th IFAC Symp. On fault Detection, Supervision and Safety for Technical Processes*, Washington, D.C., USA, June 2003. pp. 265-276.

24) Sivasubramaniam, S.V., "Reconfigurable Control Allocation Techniques for Realistic Aircraft," *Project Report*, Aalborg University Esbjerg, Denmark, 2006.

25) Bakkensen, J., Josph, V., and Merrild, U., "Flight Control Allocation Using Optimization Based Linear and Quadratic Programming," *Project Report DE7-771*, Aalborg University Esbjerg, Denmark, 2004.

26) Härkegård, Ola. "Backstepping and Control Allocation with Applications to Flight Control," *Ph. D. thesis*, Linköping University, Linköping, 2003.

27) Nocedal, J., and Wright, S. J., "Numerical Optimization," Springer, 1999.

28) Bjorck, A., "Numerical Methods for Least Squares Problems," SIAM, 1996.

29) Andrade, John M., "U.s. Military Aircraft Designations and Serials 1909-1979," Midland, 1979.