An Integrated Methodology for the Dynamic Performance and Reliability Evaluation of Fault-Tolerant Systems

Alejandro D. Domínguez-García *, John G. Kassakian, Joel E. Schindall

Laboratory for Electromagnetic and Electronic Systems
Massachusetts Institute of Technology
Cambridge, MA 02139-4307
USA

Jeffrey J. Zinchuk

The Charles Stark Draper Laboratory
Cambridge, MA 02139-3563
USA

Abstract

We propose an integrated methodology for the reliability and dynamic performance analysis of fault-tolerant systems. This methodology uses a behavioral model of the system dynamics, similar to the ones used by control engineers to design the control system, but also incorporates artifacts to model the failure behavior of each component. These artifacts include component failure modes (and associated failure rates) and how those failure modes affect the dynamic behavior of the component. The methodology bases the system evaluation on the analysis of the dynamics of the different configurations the system can reach after component failures occur. For each of the possible system configurations, a performance evaluation of its dynamic behavior is carried out to check whether its properties, e.g., accuracy, overshoot, or settling time, which are called performance metrics, meet system requirements. Markov chains are used to model the stochastic process associated with the different configurations that a system can adopt when failures occur. This methodology not only enables an integrated framework for evaluating dynamic performance and reliability of fault-tolerant systems, but also enables a method for guiding the system

design process, and further optimization. To illustrate the methodology, we present a case-study of a lateral-directional flight control system for a fighter aircraft.

Key words: Fault Tolerance, reliability, performance, Markov reliability modeling, dynamic probabilistic risk assessment, behavior, behavioral model.

1 Introduction

The safety-critical/mission-critical nature of embedded systems used in aircraft, space, tactical, and automotive applications, mandates that the systems' functionality for which they are designed be performed even in the presence of component failures. Thus, safety-critical/mission-critical systems must have the capability to adapt and compensate for component failures in a planned, systematic way [1]. These types of adaptable systems are known as fault-tolerant systems [1] .

There are well-developed techniques to support reliability evaluation of systems of moderate complexity. Among the most important ones are Reliability Block Diagrams [3], Fault-Trees [4], Dynamic Fault-Trees [5], and Markov Models [6]. The literature in the field is extensive; good references to understand the advantages and shortcomings of these techniques are [7–9]. However, all these techniques have a common shortcoming —the use of qualitative descriptions of the system's functionality to generate the system reliability model. These qualitative descriptions are usually in the form of block dia-

* Corresponding author.
E-mail address: aledan@MIT.EDU (Alejandro Dominguez-Garcia)

[1] The concept of fault-tolerance was originally formulated by Avižienis in the field of computers [2]. However, the concept of fault-tolerant systems is much broader than the computer field. There are several examples of fault-tolerant systems, e.g., aircraft, aerospace, defense, in which computers are just part of the system, but there are other equally important components or subsystems, e.g., actuators, sensors, valves, or generators. Thus, we prefer the definition given in [1], which states fault-tolerance as the ability of a system to adapt and compensate in a planned, systematic way to random failures of their components that can cause a system failure.

grams, which describe how components and subsystems are interconnected to fulfill the functions for which the system is designed. Thus, there is no quantification of the system behavior in the presence of component failure. Rather, expert judgment and previous experience is used to understand the effect of component failures on the system functionality, which is the basis to develop the reliability model. Examples of this type of analysis, where expert judgment and previous experience is used to develop the system reliability model, can be found in [10].

For conventional systems, this approach is valid, as it is possible to understand how a system can fail to perform the function for which it was designed by using this qualitative description of the system's functionality. However, this is not the case for large-complex systems or embedded software-intensive systems, which are characteristic of fault-tolerant systems [11]. For these kinds of systems, it is very difficult, if not impossible, to understand how the system behaves in the presence of hardware failures or software malfunctions without using a quantitative model of the system to quantify its behavior under failure conditions.

There have been several attempts to refine the system functional models used in reliability analysis to develop the corresponding system reliability models. In these methodologies, fault-trees are developed from architecture functional block diagrams in which local qualitative failure models are defined for each component within the system [12,13]. These component qualitative failure models are defined as collections of logical expressions, each of them describing how a component output event can result from a component internal change (a failure), or from an input event passed by another component output event. Although these works attempt to fill the gap between the system functional description (augmented with component failure descriptions) and the construction of the fault-tree, they still have several shortcomings. In these methodologies, the output deviations due to input deviations or internal failures are based on the logical expressions defined in the component local failure model. This failure model is based on the analyst's skills in understanding the

component failure behavior, and therefore it is subjective. One peculiarity of this approach is that when two components are connected, the output event of the first component must match the input event of the second component [14,15], which requires caution when defining the local failure models and limits the input deviations of a component to the set of output deviations of the previous component for cascading connections. Finally, although this approach allow the component output events to be related to simple event occurrences, it is still the task of the analyst to determine if those events can cause the system to totally fail to perform its function, or to just cause a degraded system performance which is acceptable in some situations. Thus, this does not allow us to understand the overall system performance in the presence of component failures.

In this paper, we introduce a new methodology to analyze the behavior of fault-tolerant systems in the presence of failures; minimizing the subjectivity introduced in the system analysis due to incompleteness of current reliability evaluation techniques. Rather than using a qualitative description of the system's functionality, this methodology uses a model of the system dynamics plus additional features to model component failure behavior. These features include component failure modes (and associated failure rates) and how these failure modes affect the dynamic behavior of the component. All reliability-related evaluation activities will be based on the quantitative analysis of the system dynamic performance after every possible sequence of component failures occur. This will remove the ambiguity that always arises when using qualitative models of the system's functionality (combined with expert judgment and previous experience) to develop the system reliability model. It is necessary to point out that this methodology is mainly targeted to a class of systems that can be modeled using the same formalisms used in dynamic systems and control theory, i.e, sets of differential or difference equations. It is important to note the fact that this methodology integrates system dynamic behavior and stochastic behavior due to component failures, as it generates the system reliability model from the system dynamics model, allowing the formulation of a unified model.

Section 2 of this paper is a discussion of related research. Section 3 explains in detail the elements of the methodology and lays out its mathematical foundations. Section 4 presents the main features of a MATLAB/SIMULINK® tool that supports the proposed methodology. In Section 5 a lateral-directional flight control system for a fighter aircraft is presented as a case-study. Concluding remarks and future work are presented in Section 6. Additionally, all the dynamic models and parameters of the components for the analyzed case-study presented in Section 5 are collected in several appendices at the end of the paper. The notation used is listed at the end of the paper as well.

2 Related Work

The methodology presented in this paper is related to and complements two existing frameworks independently developed in the fields of nuclear engineering and fault-tolerant computing respectively. The first one, called Dynamic Probabilistic Risk Assessment, was developed to evaluate nuclear reactors safety. The second one, called Performability analysis, was developed to analyze performance and reliability issues in degradable computing systems. In this section, we discuss the similarities and differences between the methodology proposed in this paper, and the two aforementioned methodologies.

2.1 Dynamic Probabilistic Risk Assessment

The idea of using a dynamic model of the system under control was proposed first by Amendola in 1981 to study the likelihood of accident sequences in a nuclear reactor, referred to as the Event Sequences and Consequences Spectrum (ESCS) [16]. In 1987, Aldemir [17] formalized some of the ideas introduced in [16], and introduced the idea of using a Markov model to compute the likelihood of different accident sequences in the reactor. In 1992 Devooght and Smidts laid down a rigorous mathematical formulation of the problem [18]. This methodology is commonly referred in the nuclear engineering field as

Dynamic Probabilistic Risk Assessment (DPRA).

The methodology proposed in this paper is similar to DPRA in the sense that it makes use of a dynamic (behavioral) system model, with additional information to model component failure behavior. It also makes use of a Markov chain to model the stochastic transitions between system configurations that take place when components fail. However, there are certain aspects of the kind of problems that DPRA tries to solve that makes the mathematical formulation of DPRA much more complex than the formulation of our methodology. In aircraft systems the dynamic time constants are on the order of seconds. Therefore, when a component failure occurs, the system either recovers within a short transient period, in the order of seconds, reaching a new stable steady-state (associated with a new system configuration), or it quickly becomes unstable. In this scenario, assuming failures are Poisson-distributed, the likelihood of two components failing within a time on the order of magnitude of the system dynamic time constants is negligible relative to the likelihood of just one component failing [19]. Therefore, the stochastic transitions among configurations can be modeled as a process independent of the system dynamics, and the resulting model is formulated in terms of just the probability of the various system configurations at a given time. This is not the case in nuclear power plants, where the decoupling of the stochastic transitions between configurations and system state dynamic variables cannot be done [20]. This is due to the fact that in nuclear power plants the time constants of the transitions from one configuration to another are large enough that the likelihood of another failure taking place before the system reaches a new steady-state is relevant. Therefore, in DPRA the stochastic model is formulated in terms of the probability of the system dynamic variables in a given configuration at a given time, which makes obtaining analytical solutions untractable for even simple problems.

Numerical solutions are computationally very expensive. To illustrate this, let us consider a Monte Carlo simulation, which is one of the numerical methods proposed to solve the problem [21]. With Monte Carlo, random sequences of

component failures are generated for the system operational time, and the system dynamic evolution is simulated (for each sequence of events) until meaningful reliability measures are obtained. A problem with this approach is that for very reliable systems, only a few of these simulations will lead to system failure; e.g., in a system designed to have a reliability of 10^{-6}, there will be only one simulation out of one-million simulations resulting in system failure [6]. Therefore it is necessary to carry out a larger number of simulations to obtain a meaningful reliability result. Since component failures are regarded as rare events, achieving completeness in the possible sequences of failures is difficult [22]. However, there has been some work done to try to overcome the computational problems associated with DPRA. A review of the main developments in this arena, as well as new work, can be found in [23].

2.2 Performability Analysis

In the field of fault-tolerant computing, the idea of quantifying system performance degradation in the presence of faults, and its interaction with overall system reliability measures was first addressed by Meyer [24] and Beaudry [25]. Meyer also laid down a rigorous probabilistic framework for jointly modeling performance and reliability of degradable computing systems, and coined the term *performability* to refer to this unified treatment of system performance and reliability [26]. Although there are different mathematical formalisms to model system performability, most common approaches to performability analysis are based on the use of Markov reward-models (MRM) [27,29].

The proposed methodology has similarities with performability analysis in the sense that uses Markov reward-models to define aggregated measures of system performance. A Markov reward-model is defined by a Markov chain to model the stochastic transitions between system configurations that take place when components fail, and a reward function that associates a reward to each state of the Markov chain, representing a measure of the system performance on each configuration reached after one or more components fail. For example,

system reliability can be obtained by defining the reward function as a binary function that takes value 1 whenever the system is declared as *failed* and 0 whenever it is declared as *non-failed*. Other aggregated measures can be obtained by proper definition of the reward function.

However, there is a key difference between performability and the methodology proposed in this paper, which is associated to the nature of modeling system behavior in the computer field and in the control engineering field. On one hand, the proposed methodology uses a behavioral model of the system dynamics, similar to the ones used by control engineers, to evaluate system performance, and therefore obtaining the appropriate reward function. This models of the system dynamic behavior can be defined by a set of differential equations, and expressed using a state-space representation [28]. On the other hand, when conducting performability analysis in computing systems, the performance evaluation, and the subsequent computation of reward rates, is carried out using queuing models, Markov models, or combinatorial models [27]. Thus, even if both the proposed methodology in this paper and performability analysis make use of the theory of Markov reward-models, the underlying behavioral models to obtain the reward function are very different. Therefore, the use of the proposed methodology is appropriate when the behavior of interest of the system under evaluation can be modeled with the same formalisms used by control engineers. Examples are power systems, fault-tolerant aircraft control systems, and fault-tolerant automotive control systems.

3 Methodology

The proposed methodology uses a model of the system dynamics similar to the ones proposed in switched dynamic systems theory [31]. The system model is specified by a family of continuous time subsystems, called *system configurations*, each of them defined by a set of differential equations. The dynamic model of each system configuration will depend on which components failed, how these components failed, and the order in which components failed (when

more than one failed). The switching between different configurations (starting from the nominal configuration with no failures) is governed by random failures in the system components. For each of the possible configurations, the system dynamic performance is evaluated to check out whether or not certain dynamic properties (e..g., accuracy, overshot, settling time), called *performance metrics*, meet some predefined operational requirements. Markov chains are used to model the switching process between different configurations. Once all system configurations have been evaluated, the performance metrics for each configuration and the probabilities of going from the nominal configuration to any other configuration are merged into a set of *probabilistic measures of performance*. The reader is referred to [32] for a more detailed explanation of the methodology.

3.1 System Dynamics Model

The system model emerges from the interaction of the behavioral models of each individual component within the system. We understand the term behavioral model in a similar way as understood in circuit simulation —a set of mathematical expressions that model the component external behavior (manifested through its connections to other components), without necessarily modeling the real physical processes involved in such behavior. A component behavioral model will define the component behavior not only under failure-free conditions, but also under different failure conditions (failure modes).

Components Behavioral Model

In classical reliability analysis, the concept failure mode is used to define a component operational behavior under specific internal failure conditions. Similarly, the nominal modes (no failures) of operation define the component operational behavior under failure-free conditions. Rather than making a distinction between component nominal modes and component failure modes, we introduce a concept that embraces both —component *behavioral mode*.

The behavioral modes of a component define its operational modes, for both failure-free, and internal failure conditions. Similar ideas are used in the field of fault-tolerant computing, where the the term "failure mode" is replaced by "failure semantics" [33]. However, even if the terminology used is similar, the mathematical models are not, since the mathematical formalisms used to model systems in control theory are not the same as those used in computer science.

Under the above definition of behavioral mode, a component behavioral model is defined by:

- A list of the component variables of interest for the definition of the relevant behavioral modes, and a set of mathematical expressions (behavioral equations) that constraint those variables.
- A stochastic model that describes the transitions between different behavioral modes triggered by component-internal failure conditions, or by component-external events.

As stated in Section 1, this methodology is targeted to a class of systems that can be modeled using the same formalisms used in dynamic systems and control theory. In particular, the state-space description form of a dynamic system will be used to define each mode of operation (both failure and failure-free).

Behavioral Equations

Let a component c_i have k different behavioral modes, the j^{th} behavioral mode equations $j = \{0, 1, \ldots k\}$ are defined by

$$\begin{aligned} \dot{x}_{c_i}(t) &= f^j_{c_i}(x_{c_i}(t), u_{c_i}(t)) \\ y_{c_i}(t) &= g^j_{c_i}(x_{c_i}(t), u_{c_i}(t)) \end{aligned} \tag{1}$$

where $x_{c_i}(t)$ is the vector of component state variables, $u_{c_i}(t)$ is the vector of component input variables, $y_{c_i}(t)$ is the vector of component instantaneous

outputs, $f^j_{c_i}(\cdot)$ is the component state evolution function (for the j^{th} behavioral mode), and $g^j_{c_i}(\cdot)$ is the component instantaneous output function (for the j^{th} behavioral mode).

The transitions between different behavioral modes occur stochastically, and can be triggered by internal failure conditions, or by external events. Let $U_B(t)$ be a random variable that can take values in the set $B_{c_i} = \{0, 1, \ldots k\}$, representing the component c_i instantaneous component behavioral mode. Assuming the transitions between different behavioral modes occur in a Markovian fashion, we can define the instantaneous transition rate $\lambda_{lm}(t)$ between any two component behavioral modes l and m by

$$\lambda_{lm}(t) = \frac{P(U_{c_i}(t + dt) = m \mid U_{c_i}(t) = l)}{dt} \tag{2}$$

where $l = 0, 1, \ldots k$ and $m = 0, 1, \ldots k$.

Sensor Behavioral Model Example

To illustrate the ideas introduced above, the behavioral model for a position sensor will be defined (used for example to measure the angular position of a control surface in an aircraft). In failure-free conditions, this sensor can be modeled as a first-order system with bandwidth $1/\tau_s$ (it only can measure signals up to a certain frequency). It is assumed that there is a certain latency τ_l between the sensor taking a reading and sending it out. It is also assumed that the sensor does not output continuous values of the measurements it takes, but quantized values with a resolution R_s. Finally, the measurements taken by the sensors are scaled by a constant G_s (the gain of the sensor) before they are send out.

It is assumed that the sensor has four different behavioral modes:

- **Failure-free mode (N):** bandwidth τ_s, resolution R_s, latency of τ_l, gain G_s.
- **Output-omission failure mode (O):** regardless of the sensor input read-

ing, its output is set to zero, i.e, gain 0 and other properties the same as in the failure-free mode.

- **Gain-change failure mode (G):** bandwidth τ_s, resolution R_s, latency of τ_l, gain $\hat{G}_s$.
- **Bias failure mode (B):** bandwidth τ_s, resolution R_s, latency of τ_l, gain G_s, and output biased by a factor B_s.

Under the above conditions, the behavioral equations for each operational mode of the sensor are defined by

$$U_{c_1}(t) = 0 \ (failure - free),$$
$$\dot{x}_{c_1}(t) = -\frac{1}{\tau_s}x_{c_1}(t) + \frac{1}{\tau_s}u_{c_1}(t); y_{c_1}(t) = G_s\frac{\lceil \sigma_{\tau_l} x_{c_1}(t)\frac{1}{R_s}\rceil}{\frac{1}{R_s}}; \tag{3}$$
$$U_{c_1}(t) = 1 \ (output\ omission),$$
$$\dot{x}_{c_1}(t) = -\frac{1}{\tau_s}x_{c_1}(t) + \frac{1}{\tau_s}u_{c_1}(t); y_{c_1}(t) = 0; \tag{4}$$
$$U_{c_1}(t) = 2 \ (gain\ change),$$
$$\dot{x}_{c_1}(t) = -\frac{1}{\tau_s}x_{c_1}(t) + \frac{1}{\tau_s}u_{c_1}(t); y_{c_1}(t) = \hat{G}_s\frac{\lceil \sigma_{\tau_l} x_{c_1}(t)\frac{1}{R_s}\rceil}{\frac{1}{R_s}}; \tag{5}$$
$$U_{c_1}(t) = 3 \ (bias),$$
$$\dot{x}_{c_1}(t) = -\frac{1}{\tau_s}x_{c_1}(t) + \frac{1}{\tau_s}u_{c_1}(t); y_{c_1}(t) = G_s\frac{\lceil \sigma_{\tau_l} x_{c_1}(t)\frac{1}{R_s}\rceil}{\frac{1}{R_s}} + B_s. \tag{6}$$

Transitions from the failure-free mode to each failure mode can occur, as well as transitions from the gain-change and the bias failure modes to the output-omission failure mode (operational modes do not coexist). Thus, the instantaneous transition rates are defined by

$$\begin{aligned}
\lambda_{NO}(t) &= \frac{P(U_{c_1}(t+dt) = 1 \mid U_{c_1}(t) = 0)}{dt} \\
\lambda_{NG}(t) &= \frac{P(U_{c_1}(t+dt) = 2 \mid U_{c_1}(t) = 0)}{dt} \\
\lambda_{NB}(t) &= \frac{P(U_{c_1}(t+dt) = 3 \mid U_{c_1}(t) = 0)}{dt} \\
\lambda_{GO}(t) &= \frac{P(U_{c_1}(t+dt) = 1 \mid U_{c_1}(t) = 2)}{dt} \\
\lambda_{BO}(t) &= \frac{P(U_{c_1}(t+dt) = 1 \mid U_{c_1}(t) = 3)}{dt}.
\end{aligned} \tag{7}$$

System Configurations

In a component behavioral model, a single component can exhibit different operational modes depending on its internal failure conditions and on external events. Similarly, a system will adopt different *configurations* depending on the the status of its components [23], i.e., which component operational modes are active. Thus, the system is said to be in its failure-free configuration if all its components are in their failure-free modes. The system may evolve from its failure-free configuration to another dynamic configuration if a component transitions from its failure-free mode to one of its possible failure modes. Under these conditions, a system model is defined by:

- A set of component models and how they interact with each other. Depending on the components' operational modes, the system will adopt different configurations.
- A stochastic model that describes the transitions between different system configurations triggered by component operational mode transitions.

In the previous section, we focused on defining component behavioral models for dynamic systems, adopting the state-space description for their definition. In this section, we will use the same formalism; we will assume that the systems of interest are composed of interconnected dynamic systems, and therefore a state-space description will be used to define each system configuration.

Configuration Dynamics Equations

Let k be the number of component operational mode transitions leading to the system configuration $\{i, k\}$ from the initial system configuration $\{1, 0\}$, and let $i \geq 1$ index the set of system configurations which have $k > 0$ components failed. The dynamics of the $\{i, k\}$ system configuration reached after a unique sequence of k component operational mode transitions be represented by

$$\frac{dx_s(t^{i,k})}{dt^{i,k}} = f_s^{i,k}(x_s(t^{i,k}), w(t^{i,k}))$$
$$y_s(t^{i,k}) = g_s^{i,k}(x_s(t^{i,k}), u_s(t^{i,k})) \tag{8}$$

where $x_s(t^{i,k})$ is the vector of system state variables, $u_s(t^{i,k})$ is the vector of system input variables, $y_s(t^{i,k})$ is the vector of system instantaneous outputs, $f_s^{i,k}(\cdot)$ is the system state evolution function for the $\{i,k\}$ configuration, and $g_s^{i,k}(\cdot)$ is the system instantaneous output function for the $\{i,k\}$ configuration. The time variable $t^{i,k}$ is also indexed in order to highlight the fact that two different system configurations cannot coexist, therefore their time-axis must be different.

3.2 Performance Metrics, Requirements Definition, and System Evaluation

The system evaluation process is a forward search in the sense that the evaluation starts with the system in its failure-free configuration. Then single failures are introduced in the components and the resulting system configurations are evaluated to check if certain system dynamic properties, termed *Performance Metrics*, meet some predefined criteria, termed *Performance Requirements*. The configurations reached after a sequence of failures that do not meet the performance requirements are declared as *failed*, and no other system configurations can be reached from them. The configurations declared as *non-failed* meet the performance requirements, meaning that the system is still operational and can still perform the function for which it was designed. Other system configurations (after subsequent component failures) can be reached from them.

Performance Metrics

Performance metrics are a set of m system-related properties, denoted by $Z_1, Z_2, \ldots, Z_m$, that, for each system configuration $\{i,k\}$, can be computed by using the system dynamic equations (8). Performance metrics will quantify how well a system performs the function for which it was designed.

The performance metrics chosen to evaluate a system depend on the nature of the system. For example, in a tracking system, dynamic-related properties may be important, e.g., the tracking error; the overshoot; the settling time; or the poles and zeros locations if the system is linear. If the system to be evaluated is a power system, performance metrics of interest may be the instantaneous, or the average power consumption within the system; the maximum power delivered by a single power element; or the maximum current flowing through some components.

All the properties mentioned above, are usually taken into account in the design and evaluation of any conventional system, which are usually evaluated for the system nominal configurations (components nominal behavioral modes). However, one of the purposes of a fault-tolerant system is to be able to accommodate component failures and keep delivering its function. Thus, in this methodology the performance metrics are not only evaluated for the system nominal configuration, but also for each configuration with failed components, that can be reached from the system nominal configurations.

Performance Requirements

Performance requirements are defined through sets $\Phi^{i,k}_{Z_1}, \Phi^{i,k}_{Z_2}, \ldots, \Phi^{i,k}_{Z_m}$, and represent the values that the performance metrics are allowed to take on each *non-failed* system configuration. For each system configuration $\{i, k\}$ reached after a sequence of failures of size k, the values of the performance metrics $Z_1, Z_2, \ldots, Z_m$ will be checked to see if they are within the predefined performance requirements $\Phi^{i,k}_{Z_1}, \Phi^{i,k}_{Z_2}, \ldots, \Phi^{i,k}_{Z_m}$.

For example, let $\|\cdot\|$ be a norm in $\mathbb{R}^n$. Let $c_j \in \mathbb{R}^{n_j}$ and $r_j > 0$, then the requirements for the j^{th} performance metrics Z_j are defined by the set

$$\Phi^{i,k}_{Z_j} = \{z \in \mathbb{R}^{n_j} \mid \| z - c_j\| \leq r^{i,k}_j\}. \tag{9}$$

The choice of c_j and $r^{i,k}_j$ depend on the system under evaluation, i.e., the

functionality the system is designed for and the performance requirements imposed on that functionality. For example, as it will be further illustrated in Section 5, when evaluating the performance of an aircraft, Z_j could represent the aircraft sideslip; and c_j could be defined as the aircraft sideslip in the failure-free condition. Thus, equation 3.2 would represent the maximum deviation allowed (represented by $r_j^{i,k}$) on the aircraft sideslip in any particular configuration with respect to its nominal (failure-free) sideslip. If the system to be evaluated is a tracking system, Z_j could be the system response, c_j the reference, and $r_j^{i,k}$ the maximum allowed tracking error.

Dynamic Performance Evaluation

For each system configuration given by $\{f_s^{i,k}, g_s^{i,k}\}$, the system behavior is analyzed in order to quantify its performance metrics. A given system configuration will declared as *failed* if any of the system performance metrics does not meet its requirements, otherwise the configuration is declared as *non-failed*.

Let $Z_1, Z_2, \ldots Z_m$ be the performance metrics of a system, which can take values $z_1^{i,k}, z_2^{i,k}, \ldots z_m^{i,k}$ for each system configuration $\{i,k\}$. Let $s_{i,k}(t^{i,k})$ be an indicator variable that takes value 1 when the configuration $\{i,k\}$ is declared as *non-failed*, and 0 otherwise, then

$$s_{i,k}(t^{i,k}) = \begin{cases} 1, & \text{if } z_j^{i,k} \in \Phi_{Z_j}(i,k) \ \forall \ j = 1,2,\ldots m \\ 0, & otherwise. \end{cases} \tag{10}$$

3.3 System Stochastic-Behavior Model

The transitions between configurations occur stochastically and are triggered by random behavioral mode transitions within the system components. As mentioned in section 3.1, each component can have different behavioral modes, and thus transition rates between different modes must be defined. We assume

that the system evolves between configurations in a Markovian fashion, i.e., the system configurations at future times will only depend on the configuration at the present time. Therefore the Chapman-Kolmogorov equations can be used to compute the probabilities of each system configuration status (failed or non-failed).

Let $\lambda_{j,k-1}^{i,k}$ be the transition rate (associated with the transition rate between two component operational modes) that causes the system to go from configuration $\{j, k-1\}$ to configuration $\{i, k\}$, and $\lambda_{i,k}^{m,k+1}$ be he transition rate that causes the system to go from configuration $\{i, k\}$ to configuration $\{m, k+1\}$. Let $X(t)$ denote the system configuration at time t. Let $p_{i,k}(t)$ be the probability that, at any time $t \geq 0$, the configuration $\{i, k\}$ is declared as *non-failed*, conditioned on the fact that the system was in the failure-free configuration $\{1, 0\}$ at $t = 0$ (with probability 1). Then

$$\begin{aligned} \frac{dp_{i,k}(t)}{dt} &= \lambda_{j,k-1}^{i,k}(t)p_{j,k-1}(t) - \sum_{m} \lambda_{i,k}^{m,k+1}(t)p_{i,k}(t) \\ p_{i,k}(t) &= P(X(t) = \{i, k\}, s_{i,k}(t^{i,k}) = 1 | X(0) = \{1, 0\}). \end{aligned} \tag{11}$$

Similarly, let $p_{i,k}^*(t)$ be the probability that, at a time $t \geq 0$, the configuration $\{i, k\}$ is declared as *failed*, conditioned on the fact that the system was in the failure-free configuration $\{1, 0\}$ at $t = 0$. Then

$$\begin{aligned} \frac{dp_{i,k}^*(t)}{dt} &= \lambda_{j,k-1}^{i,k}(t)p_{j,k-1}(t) \\ p_{i,k}^*(t) &= P(X(t) = \{i, k\}, s_{i,k}(t^{i,k}) = 0 | X(0) = \{1, 0\}). \end{aligned} \tag{12}$$

The choice of (11) or (12) to represent the stochastic behavior of configuration $\{i, k\}$ will depend on the dynamic performance evaluation resulting from (10). However, as mentioned in the introduction, and shown in (11) and (12), the formulation of the stochastic model of the transitions between configurations is independent of the system dynamic variables, and, therefore, it is possible to compute the configuration probabilities $p_{i,k}(t)$ independently of the system dynamics state-variables $x_s(t)$.

The state-transition matrix associated with the Markov model that governs the transitions between all system configurations is obtained by assembling, for each configuration $\{i, k\}$, equations (11) or (12). Let A be the state-transition matrix associated with the Markov model that governs the transitions between the set of reachable system configurations, then the system configurations probability vector $P(t)$, can be computed by solving

$$\begin{aligned} \frac{dP(t)}{dt} &= AP(t) \\ P(0) &= [1\ 0\ 0\ ...\ 0]'. \end{aligned} \tag{13}$$

3.4 Probabilistic Measures of Performance and Reliability

The performance metrics $Z_1, Z_2, \ldots Z_m$ are useful in determining whether each individual system configuration is declared as *failed* or *non-failed*, and the Markov model given by (13) allows the probabilities of declaring each configuration as *non-failed* or *failed* to be computed. However, it is necessary to define other sets of measures to quantify the system as a whole, i.e., aggregated measures for all the possible system configurations. System Reliability R and unreliability Q are examples of these aggregated measures. The definition of these aggregated measures is very important since they will be used to compare different system architectures, and to find weak points in a design.

In order to define these aggregated measures, we will make use of Markov reliability models (MRM), which have been used extensively to quantify the performability of computer systems as mentioned in Section 1. Every aggregated measure, including System Reliability and Unreliability, will be derived from the general MRM formulation [27], which is defined by:

(1) a Markov chain $Y = \{Y(t) : t \geq 0\}$, indexed in time by $[0, \infty)$, and taking values in some countable set $C = \{1, 2, \ldots N\}$; and
(2) a reward function $r : C \rightarrow \mathbb{R}$, where the reward associated with each $i \in C$ is denoted by $r(i)$.

System reliability R can be obtained by using the indicator function defined in (10) (which takes value 1 when the configuration $\{i,k\}$ is declared as *non-failed*, and 0 otherwise) to define the reward model, thus $r_S(i,k) = s_{i,k}$. Then, system reliability is computed as

$$R = \mathbb{E}(r_S) = \sum_{i,k} r_S(i,k) p_{i,k}(t). \tag{14}$$

Similarly, system unreliability Q can be computed by defining the reward function $r_{\overline{S}}(i,k) = 1 - s_{i,k}$, thus

$$Q = \mathbb{E}(r_{\overline{S}}) = \sum_{i,k} r_{\overline{S}}(i,k) p_{i,k}(t). \tag{15}$$

Aggregated measures of performance can also be computed for each performance metric Z_j, with $j = 1, 2, \ldots m$, by defining a reward function as

$$r_{Z_j}(i,k) = h_j(Z_j) \;\forall\; j = 1, 2, \ldots m \tag{16}$$

where $h_j(\cdot)$ is a real function. For example, if the Z_j performance metric considered is the electrical power consumption, it is possible to obtain the system average power consumption among all possible *non-failed* operational conditions by defining $r_{\overline{P}}(i,k) = z_j(i,k)$ when $s_{i,k}(t^{i,k}) = 1$, and $r_{\overline{P}}(i,k) = 0$ when $s_{i,k}(t^{i,k}) = 0$; thus the average power consumption $\overline{P}$, when the system is in a *non-failed* configuration, can be computed as

$$\overline{P} = \mathbb{E}(r_{\overline{P}}) = \sum_{i,k} r_{\overline{P}}(i,k) p_{i,k}(t). \tag{17}$$

4 Matlab/Simulink Tool

In order to automate the proposed methodology, we developed a MATLAB/ SIMULINK® based tool –InPRESTo, an acronym for Integrated Performance and Reliability Evaluation SIMULINK® Toolbox. The reader is referred to [32,34] for a more detailed description of the tool functionality. The toolbox helps the analyst to:

(1) thoroughly evaluate the system behavior in the presence of component failures;
(2) evaluate the effectiveness of failure detection, isolation, and reconfiguration mechanisms (FDIR);
(3) uncover weak points in the design, i.e., single points of failure and common modes of failure;
(4) quantify the main contributors to system unreliability;
(5) quantify the advantages of using an specific architecture among different alternatives.

The basic functionality of InPRESTo is displayed in Fig. G.1. The inputs to InPRESTo are:

- The system dynamics behavioral model defined in the SIMULINK® environment. The component failure behavior can be built into each component model by *"drag and drop"* from a SIMULINK® library called *Failure Models*. This library can be accessed from the SIMULINK® GUI, and contains several failure models.
- System performance metrics and their requirements, which are defined within the SIMULINK® system behavioral model. There is a library called *Performance Metrics* with predefined performance metrics models from which the analyst can *"drag and drop"* as well.
- Evaluation parameters, which are additional parameters needed to run the analysis.

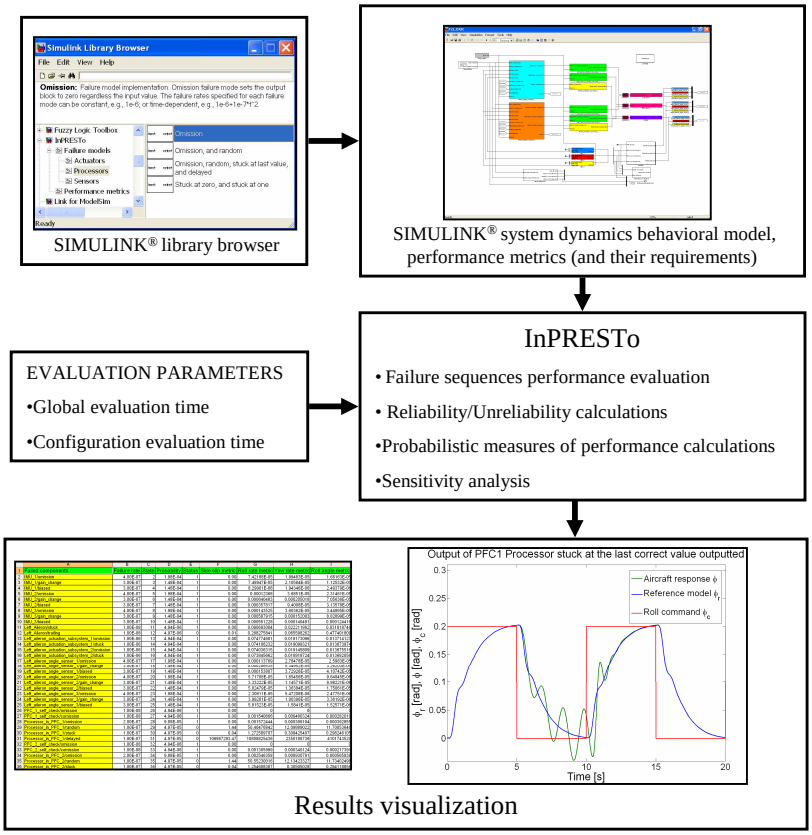


Fig. 1. Basic functionality of InPRESTo.

The evaluation engine used by InPRESTo is described in [32]. It was coded as a collection of MATLAB® functions. The tool can automatically perform exhaustive analyses of all possible sequences of component failures that can yield a system failure, or the analysis can be truncated when sequences of component failures of a certain size are reached. In the latter case, bounds on the reliability and unreliability are calculated to estimate the error introduced by truncating the analysis. InPRESTo can also calculate probabilistic measures of performance.

All the analysis results are automatically collected in several excel spreadsheets, and for each system configuration, the tool will output:

- The sequence of component failures leading to that system configuration.
- Probability of being in each system configuration
- The performance metrics values in that system configuration.
- A tag to indicate whether the configuration is failed or not.

Additionally, System reliability and unreliability values are collected, as well as the probabilistic measures of performance associated with each performance metric. The tool also allows each particular system configuration to be simulated individually.

5 Lateral-Directional Flight Control System Case-Study

In this section, a case-study for the lateral-directional flight control system of a fighter aircraft [35] is presented. The authors want to clarify that the architecture redundancy schemes and FDIR mechanisms used in this case-study are by no means novel. The purpose of this case-study is not to introduce new architectural concepts for fault tolerant avionics. The purpose of this case-study is to show how the methodology presented in this paper (and its supporting MATLAB/SIMULINK® tool) enables a completely new way of analyzing fault-tolerant avionics systems. In this regard, it illustrates how this methodology can be applied to analyze existing systems, or it can be used during the design phase to evaluate a proposed system architecture; identify its weak points; improve the architecture in several iterations by removing those weak points, and at the same time improving the aircraft dynamic performance under failure conditions; and finally compare the trade-offs between the different design iterations.

To illustrate this process, three architectural alternatives will be explored. In the first one, called dual channel architecture (DCa), only pure-redundancy is used as a vehicle to achieve fault-tolerance. The analysis results of this first alternative will show that even in the presence of redundancy, single fault-tolerance is not achieved. Based on the results analysis of the DCa design, a second design, called enhanced dual channel architecture (EDCa) will be introduced. The analysis of this second architecture will uncover some weak points, which will be addressed by a third iteration of the design, called dual-dual channel architecture (DDCa). Finally, the three architectural alternatives will be compared in terms of complexity, cost, overall performance, reliability,

and single fault tolerance.

System Performance Metrics Definition and Associated Requirements

The first step in carrying out the analysis of any system is to establish its performance metrics and associated requirements. In this case, the performance metrics chosen are the aircraft state variables: the sideslip $\beta(t)$, the body axis roll rate $p_b(t)$, the body axis yaw rate $r_b(t)$, and the body axis roll angle $\phi(t)$. Thus:

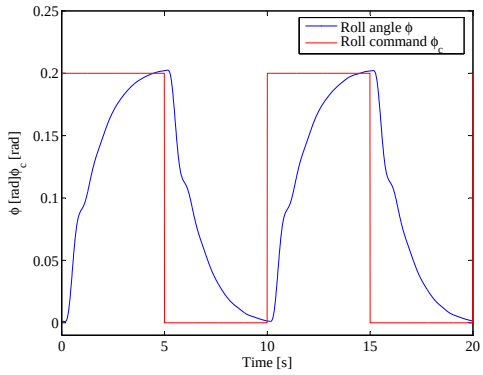
$$Z_1 = \beta(t), \tag{18}$$
$$Z_2 = p_b(t), \tag{19}$$
$$Z_3 = r_b(t), \tag{20}$$
$$Z_4 = \phi(t). \tag{21}$$

The dynamic behavior of a reference aircraft [35] (see Appendix G for the model details) will be used to define the performance metrics requirements. This reference aircraft state variables are denoted by the sub index $_r$, i.e., the sideslip is denoted by $\beta_r(t)$, the body axis roll rate by $p_{b_r}(t)$, the body axis yaw rate by $r_{b_r}(t)$, and the body axis roll angle by $\phi_r(t)$. Thus, the performance metrics requirements are defined as

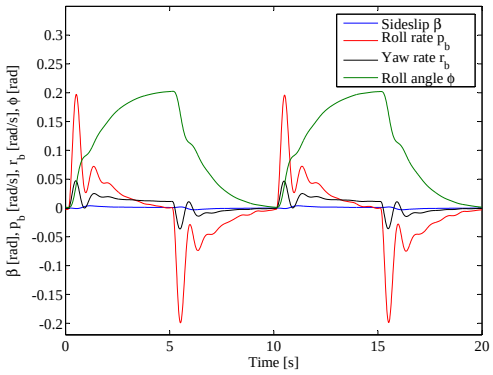
$$\Phi_{Z_1} = \{\beta(t) \in \mathbb{R} \mid \| \beta(t) - \beta_r(t)\|_\infty \leq r_\beta\}, \tag{22}$$
$$\Phi_{Z_2} = \{p_b(t) \in \mathbb{R} \mid \| p_b(t) - p_{b_r}(t)\|_\infty \leq r_{p_b}\}, \tag{23}$$
$$\Phi_{Z_3} = \{r_b(t) \in \mathbb{R} \mid \| r_b(t) - r_{b_r}(t)\|_\infty \leq r_{r_b}\}, \tag{24}$$
$$\Phi_{Z_4} = \{\phi(t) \in \mathbb{R} \mid \| \phi(t) - \phi_r(t)\|_\infty \leq r_\phi\}, \tag{25}$$

where $r_\beta = 0.15$rad, $r_{p_b} = 0.45$rad/s, $r_{r_b} = 0.45$rad/s, and $r_\phi = 0.15$rad.

The response of the aircraft reference model for a 0.2rad, 0.1Hz square wave in the roll command ϕ_c is displayed in Fig. G.2. This response will be used through the remaining of the case-study to compare it with the actual aircraft performance for different failures.



(a) Roll command ϕ_c, and corresponding roll angle response ϕ.



(b) Sideslip response β, roll rate response p_b, yaw rate response r_b, and roll angle response ϕ.

Fig. 2. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in the roll command ϕ_c.

5.1 Dual Channel Architecture

The proposed dual channel architecture (DCa), Fig. G.3, is based on the use of pure redundancy. No FDIR mechanisms are implemented, except for voting algorithms for the triple redundant measurements. The architecture is composed of two redundant primary flight computers (PFC_1 and PFC_2) that receive information about aircraft attitude from three redundant inertial measurement units (IMU_1, IMU_2 and IMU_3) cross strapped to both computers; and also information of the control surface position from triple redundant position sensors for rudder (RPS_1, RPS_2 and RPS_3), left aileron ($LAPS_1$, $LAPS_2$ and $LAPS_3$), and right aileron ($RAPS_1$, $RAPS_2$ and $RAPS_3$) also cross strapped to both computers. Both PFCs have a voting algorithm implemented to compute the actual aircraft attitude from the triple redundant IMUs measurements, and voting algorithms for each set of triple LAPSs, RAPSs, and RPSs measurements. Both PFCs have also implemented control laws that, compute the appropriate commands for the control surface actuation subsystems, based on the IMUs measurements and the pilot inputs through the stick and the pedals. Each control surface is actuated by two redundant actuation subsystems, $LAAS_1$, and $LAAS_2$ for the left aileron; $RAAS_1$, and $RAAS_2$ for the right aileron; and RAS_1, and RAS_2 for the rudder. The outputs of each pair of actuation subsystems are mechanically combined to produce the appro-

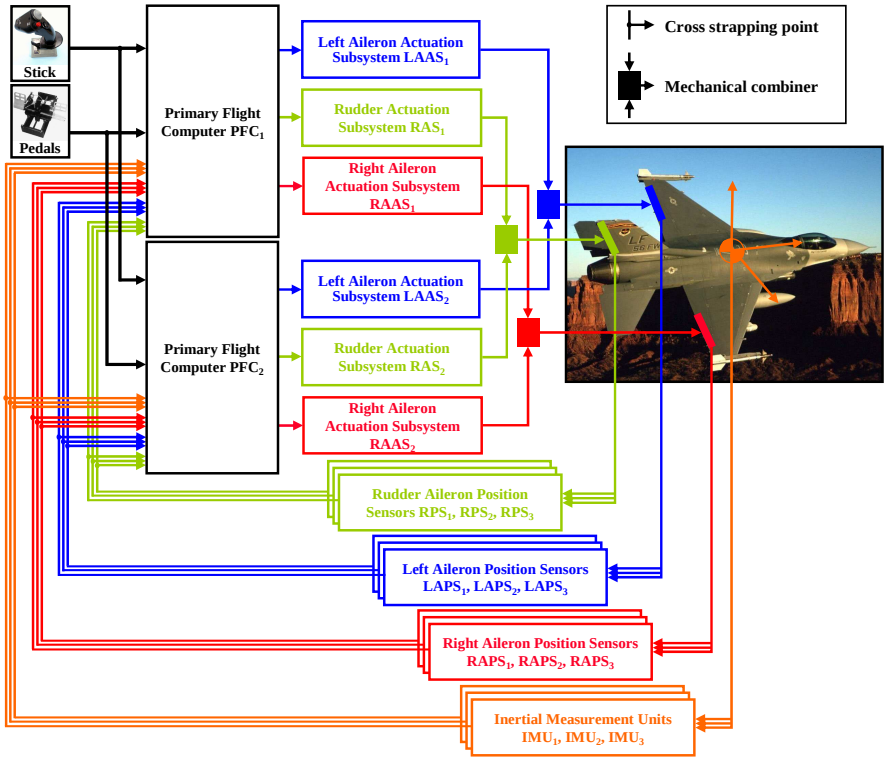


Fig. 3. **Lateral-Directional Flight Control System Fault-Tolerant Architecture.**

priate command for the corresponding control surface. Each element of the control-surface-actuation-subsystem pair is commanded independently from each PFC. Therefore, there are three actuation subsystems per PFC, commanding both left and right ailerons, and the rudder. Each actuation subsystem is composed of a current-controlled electric motor. When any of the above mentioned hardware components fail, it remains in the control loop, and the additional redundant units are supposed to compensate for the failure.

The component behavioral models for each hardware and software component, i.e., primary flight computers (PFC), voting algorithms, control laws, inertial measurements units (IMU), rudder position sensors (RPS), left and right aileron position sensors (LAPS and RAPS), rudder actuation subsystems (RAS), left and right aileron actuation subsystems (LAAS and RAAS) rudder (Ru), left aileron (LA), and right aileron (RA) are described in Appendices A-F. Table 1 collects information corresponding to the failure models of the different hardware components. The possible failure modes of each component

Table 1
Component failure model parameters for the dual channel architecture.

Component	*Failure modes*	*Description*	U_B	$\lambda(/h)$
PFC_1, PFC_2	Omission	Output set to zero	1	$2 \cdot 10^{-7}$
	Random	Random output between -5 and 5	2	10^{-7}
	Stuck	Output stuck at last correct value	3	10^{-7}
	Delayed	Output delayed 0.2 s	4	10^{-7}
$LAAS_1$, $LAAS_1$, $RAAS_1$	Omission	Output set to zero	1	10^{-6}
$RAAS_2$, RAS_1, RAS_2	Stuck	Output stuck at last correct value	2	10^{-6}
Ru, LA, RA,	Omission	Output set to zero	1	10^{-8}
	Trailing	Output commanded by the aircraft dynamics	2	10^{-8}
IMU_1, IMU_2, IMU_3	Omission	Output set to zero	1	$4 \cdot 10^{-7}$
	Gain change	Output scaled by a factor of 1.5	2	$3 \cdot 10^{-7}$
	Biased	Output biased by a factor of 0.3	3	$3 \cdot 10^{-7}$
$LAPS_1$, $LAPS_2$, $LAPS_3$,	Omission	Output set to zero	1	$4 \cdot 10^{-7}$
$RAPS_1$, $RAPS_2$, $RAPS_3$,	Gain change	Output scaled by a factor of 1.5	2	$3 \cdot 10^{-7}$
RPS_1, RPS_2, RPS_3	Biased	Output biased by a factor of 0.3	3	$3 \cdot 10^{-7}$

are listed in column 2 of Table 1, while column 3 is an explanation of the effect of each behavioral mode on the component behavior. U_B in column 4 is the variable that assigns the corresponding failure mode to the component behavioral model equations (see Appendices A-F). The last column of Table 1 collects the failure rates λ associated with each failure mode, necessary to build the state-transition matrix associated to the Markov reliability model.

To complete the system behavioral model, a linear lateral-directional aircraft dynamic model [36], [37] interacting with the avionics architecture model described above is included. The state variables of this model are the sideslip β, the body axis roll rate p_b, the body axis yaw rate r_b, and the body axis roll angle ϕ. The control surface commands are both left and right aileron angles δ_a^l and δ_a^r, and the rudder angle δ_r. The complete state-space representation of the aircraft lateral-directional dynamics is shown in Appendix G.

Performance and Reliability Evaluation

The system evaluation was carried under specific conditions. The aircraft is considered to be in a cruising phase with forward velocity $V = 178$m/s, pitch

Table 2
Dual channel architecture: single points of failure and unreliability for different levels of truncation and an evaluation time of 500 h.

Truncation level	*Unreliability lower bound*	*Unreliability upper bound*	Single points of failure	*# of system configurations*
2	$5.12 \cdot 10^{-4}$	$5.82 \cdot 10^{-4}$	11	64
3	$5.20 \cdot 10^{-4}$	$5.20 \cdot 10^{-4}$	11	3088

angle $\alpha_0 = 0.216$rad and and a cruising altitude of $10,668$m.

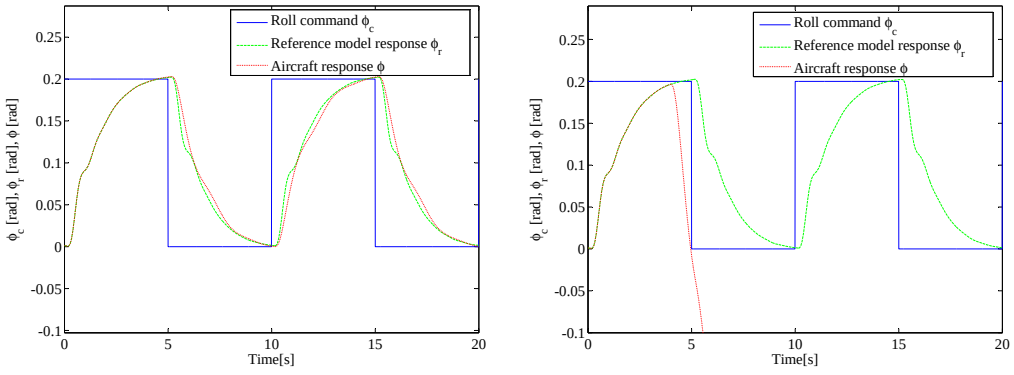
Truncation techniques were used to avoid the state-space explosion of the Markov model [38,39]. The maintenance period of the aircraft $T = 500$h was considered to compute the system unreliability estimates. Table 2 shows the probability of system failure (unreliability) at the end of the maintenance period for different levels of truncation. It can be seen that it is enough to evaluate up to system configurations with three components failed (the upper and lower reliability estimates coincide). Truncating after three component failure events yields a system unreliability of $5.20 \cdot 10^{-4}$. There is a trade-off between achieving a higher accuracy in estimating reliability and computational time for performing the evaluation. By truncating the evaluation at the second level, only 64 system configurations are evaluated, and the evaluation takes less than 4 minutes. If the truncation is done at the third level, 3088 possible configurations are analyzed and the evaluation takes 2 hours and 46 minutes. The computation was carried out on a machine with a 2.1 GHz Pentium® M processor, and 1.5Gb of RAM. It is important to note that with the proposed methodology, only 3088 simulations were needed to obtain a good reliability estimate.

For clarity, in the remaining of the results analysis, only the behavior of one performance metric — the aircraft roll angle ϕ, will be analyzed. Thus, for several single failures, the aircraft roll angle response ϕ will be plotted together with the reference model response ϕ_r, for a 0.2rad, 0.1Hz square wave in roll command ϕ_c.

Single aileron failures

Figure 4(a) shows the roll angle aircraft response ϕ, and reference model response ϕ_r, for a single failure in the left aileron, in which it fails by getting stuck at the position it was in when the failure occurred. Although the system performance is degraded, the performance metrics (i.e., the aircraft state variables) remain within their requirements. Fig. 4(b) shows the aircraft response for another failure of the left aileron. The failure mode is such that the aileron is now commanded by the aircraft dynamics, i.e., the aileron trails. In this case, the failure is catastrophic. It can be seen that 4s after the failure occurs, ϕ rapidly increases. This means that the aircraft is rolling without any control.

In these cases, there are not many things that can be done to improve the system performance (in the stuck-failure-mode), or to keep the aircraft stable (in the trailing-failure-mode), since the aileron is non-redundant, and when it fails, it affects the aircraft aerodynamics. However, as reported in [40], NASA developed a system, called propulsion controlled aircraft (PCA), to compensate for failures in the control surface by reconfiguring the engine thrust control system in order to use differential thrust to maneuver the aircraft. This could be an example on how to achieve fault-tolerance in a system in which it is not possible to use redundancy.



(a) Stuck failure mode. (b) Trailing failure mode.

Fig. 4. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in the left (or right) aileron, and a failure injection time $t_f = 4$ s.

Single aileron actuation subsystem failures

For any of the left aileron actuation subsystems, Fig. G.5 displays the aircraft behavior for a failure-by-stuck, i.e, the actuation subsystem acts as a load of constant torque for the remaining healthy actuation subsystem. In this condition, the roll angle aircraft response ϕ does not perfectly match the reference model response ϕ_r; thus a degraded performance behavior results. Nevertheless the system is stable and the performance metrics lie within the requirements. Although not displayed here, a similar effect on the aircraft dynamics occurs when one aileron actuation subsystem fails by omission. Therefore the resulting configurations are both declared as non-failed. In this case, having pure redundancy is enough to compensate for any failure in the left (or right) aileron actuation subsystems.

However, there are ways to improve the performance of the aircraft when failures in the left (or right) aileron actuation subsystems occur. An example could be to introduce some sort of FDIR mechanism to detect and isolate (some, or all) failures within the actuation subsystem; and if necessary, reconfigure the control laws in the remaining actuation subsystem to account for those failures.

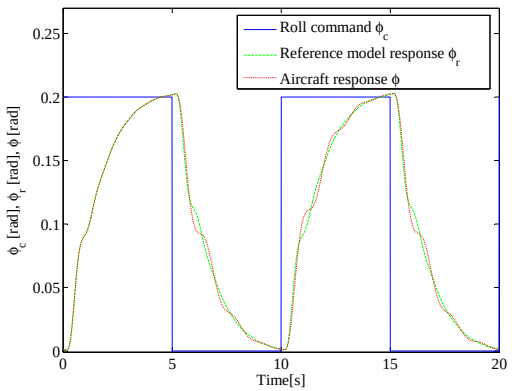
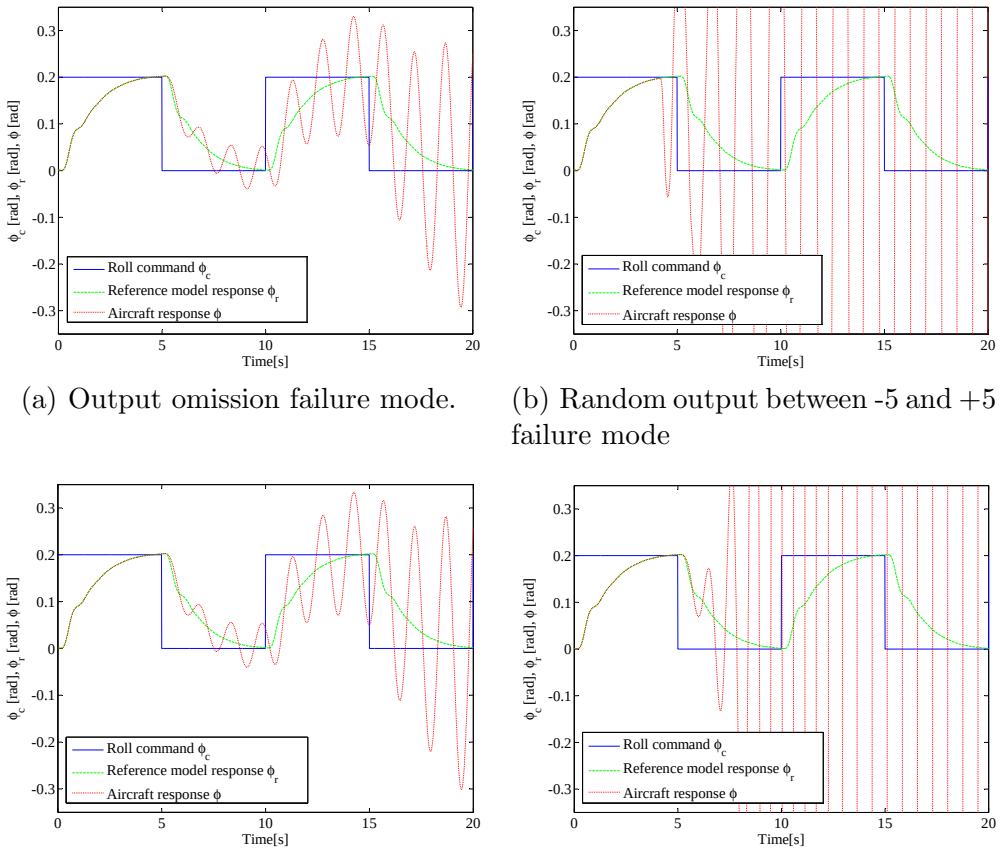


Fig. 5. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for a failure-by-stuck in one of the left (or right) aileron actuation subsystems, and a failure injection time $t_f = 4$ s.



(a) Output omission failure mode.

(b) Random output between -5 and +5 failure mode

(c) Output stuck failure mode.

(d) Output delayed failure mode.

Fig. 6. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in one of the primary flight computers, and a failure injection time $t_f = 4$ s.

Single primary flight computer failures

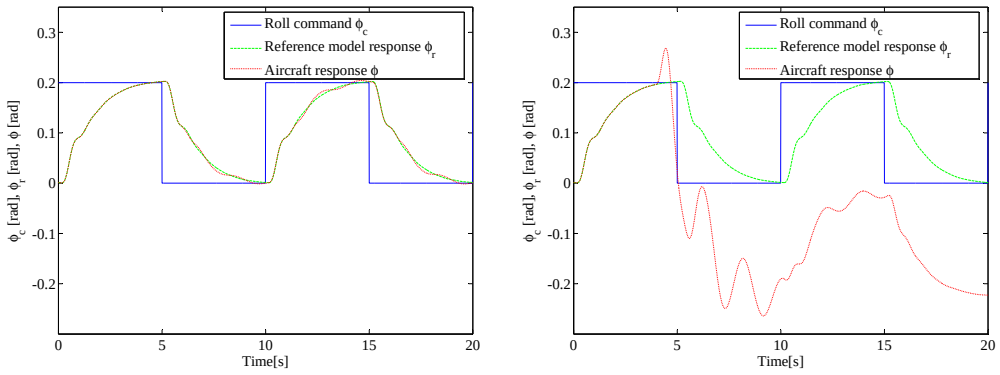
Figure G.6 shows the aircraft behavior for different failure modes in any of the primary flight computers. It can be seen that, despite the presence of another primary flight computer in command of half of the system, any failure in one of the computers will cause the aircraft to become unstable. For a failure-by-output-omission, Fig. 6(a), one of the channels in the forward loop (Primary flight computer, left and right aileron actuation subsystems, and rudder actuation subsystems) is effectively removed, i.e., the computer stops sending any command to the actuation subsystems, therefore these stop commanding the control surfaces. This result in an alteration of the closed-loop dynamics that makes the system become unstable. In the case of a linear system, the

effect of removing one channel of the control-loop would result in the relocation of the system poles, some of them moving to the right-half-plane, which would make the system unstable. A similar explanation can be given for the failure-by-stuck-output, 6(b). In this case, the computer output is set to a constant, therefore the control surface actuation subsystems commanded by the computer set their outputs to a constant value, acting as a load for the actuation subsystems in the other channel, but causing the same effect as before: an alteration of the closed-loop dynamics that makes the system become unstable. The effect of the other two failure modes is even more dramatic as can be seen in Fig. 6(c) for the failure-by-delayed-output, and in Fig. 6(d) for the failure-by-random-output.

The most important conclusion extracted from the effect of primary flight computer failures on the aircraft response is that using redundancy alone is not sufficient to achieve fault-tolerance, unlike the case of failures in the control surface actuation subsystems, where redundancy alone was sufficient. The results analysis also point to possible solutions to overcome the problems shown. First, failure detection and isolation is not enough; reconfiguration of the control laws in the second computer is also necessary. The reason for this can be understood if the failure-by-output-omission is analyzed; the effect of this failure on the system behavior is equivalent to the effect of detecting any failure in the primary flight computer and isolating the failure by shutting the computer down. From Fig. 6(a), it is enough to understand that this strategy will not work, and it is necessary to do something else: to reconfigure the control laws in the remaining computer after the failed computer is shut down.

Single rudder failures

Figure G.7 shows the aircraft response for failures in the rudder. Similarly to the aircraft behavior displayed for failures in the left (or right) ailerons, Fig. 7(a) corresponds to a failure-by-stuck of the rudder, which degrades the aircraft performance, but the performance metrics are still within require-



(a) Stuck failure mode. (b) Trailing failure mode.

Fig. 7. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in the rudder, and a failure injection time $t_f = 4$ s.

ments. Fig. 7(b) corresponds to a trailing failure of the rudder, which results in a a system failure.

As mentioned for failures in left and right ailerons, this is not a problem that can be solved using redundancy, since there is only one rudder in the aircraft, but it could be solved using the propulsion controlled aircraft approach already mentioned, and reported in [40].

Single rudder actuation subsystem failures

Figure G.8 shows the effect of a failure-by-stuck in any of the rudder actuation subsystems. A similar behavior is obtained for a failure-by-omission in one of the rudder actuation subsystems cause in the system: a degraded performance, but not instability.

As mentioned in the case of failures in the left or right ailerons, the aircraft performance could be improved by introducing failure detection, isolation, and reconfiguration (FDIR) mechanisms to — partially or completely — detect and isolate failures within the actuation subsystem; and if necessary, reconfigure the control laws in the remaining actuation subsystem to account for those failures.

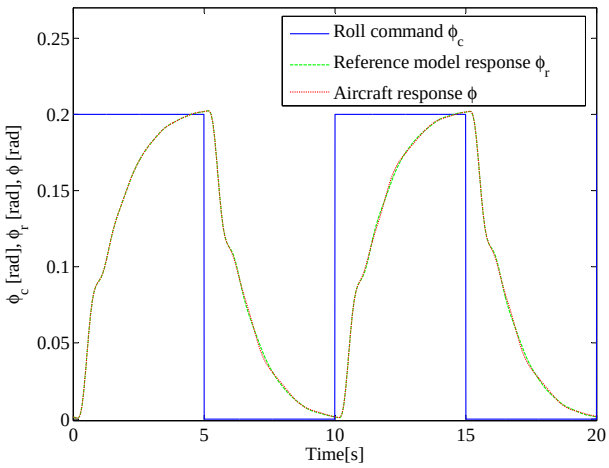


Fig. 8. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for a failure-by-stuck in one of the rudder actuation subsystems, and a failure injection time $t_f = 4$ s.

5.2 Enhanced Dual Channel Architecture

In the dual channel architecture presented in Section 5.1, every component, except the control surfaces, is duplicated or triplicated in order to achieve fault-tolerance. The detailed analysis of this architecture shows that redundancy was not enough to achieve fault-tolerance in some cases. In this section, and based on the results analysis of the dual channel architecture, an enhanced dual channel architecture is proposed.

The main conclusions extracted from the analysis of single failures in the dual channel architecture are:

(1) Some failures in the control surface cause the system to become unstable and cannot be overcome using conventional fault-tolerant techniques.
(2) Failures in the control surface actuation subsystem cause degraded performance but do not cause the aircraft to become unstable.
(3) Despite the presence of two primary flight computers, any single failure in a computer will cause the aircraft to become unstable.

As mentioned before, it may be possible to solve each of the problems listed above. However, the purpose of this paper is to show how the methodology

can be used to uncover weak design points, and how they can be improved, it is not the purpose of this paper to design sophisticated FDIR mechanisms, but only to illustrate how the effectiveness of these can be tested. Therefore, only the issues related to the third item listed above, i.e., how to compensate failures in any of the PFCs will be treated.

Focusing on the PFC failures, from the results analysis of the dual channel architecture, it was concluded that failure detection and isolation is not sufficient, reconfiguration of the control laws in the remaining computer is also necessary. As explained before, the effect of a failure-by-output-omission in the computer is equivalent to the effect of detecting any failure in the primary flight computer and isolating the failure by shutting the computer down. Figure 6(a) shows that this strategy will not work, and it is necessary to reconfigure, as well, the control laws in the remaining computer after the failed computer is shut down.

The proposed enhanced dual channel architecture (EDCa) is very similar to the dual channel architecture presented in, Fig. G.3) in the sense that it has the same components connected in a very similar way. The three main differences are: Within each PFC, there is a failure self-detection circuit (PFC_1-SD and PFC_2-SD) that will be the core of the FDIR mechanism described later. Each PFC exchanges information of its status (failed or operational) with the other PFC. The control laws within the processors of each PFC are reconfigurable depending on the status of the other PFC. These additional components and features allow the implementation of an FDIR mechanism:

- **Detection.** The self-detection circuit PFC-SD implemented in each PFC checks the range of the signals outputted by the PFC processor, and if they are within a certain range, then they are considered as valid, otherwise the self-detection circuit reports a failure. Additionally, the rate of change of the outputted signals is checked, if the self-detection circuit detects no rate of change, then a failure is reported.
- **Isolation.** Once the self-detection circuit detects a failure, the main PFC processor is shut down.

- **Reconfiguration.** Once the self-detection circuit detects a failure, a reconfiguration signal is sent to the remaining PFC to double the gain of the control surface actuation subsystems controllers. This will compensate for the fact that only the control surface actuation subsystems commanded by the remaining computer are operational.

The proposed FDIR candidate should handle failure-by-output-omission, failure-by-stuck-output, and failure-by-random-output. It is not clear by this method whether failure-by-delayed-output will be handled effectively by this FDIR candidate. This will be explored further in the next section.

Table 3
Enhanced dual channel architecture: primary flight computers self-detection circuits failure model parameters.

Component	*Failure modes*	*Description*	U_b	$\lambda(/h)$
PFC_1-SD, PFC_2-SD	Omission	Output set to zero regardless the input	1	10^{-8}
	Commission	Output set to one regardless the input	2	10^{-8}

Table 3 collects the information corresponding to the failure models of the PFC-SDs introduced in the enhanced dual channel architecture — the self-detection circuits for each PFC. The failure models parameters for the rest of the components are the same as for the pure-redundancy architecture, Table 1.

Performance and Reliability Evaluation

The conditions under which the enhanced dual channel architecture was evaluated are the same as the ones used to evaluate the pure-redundancy architecture, i.e., the aircraft cruising at an altitude of $10,668$m with with forward velocity $V = 178$m/s, pitch angle $\alpha_0 = 0.216$rad, and the same configuration evaluation time ($t_c = 20$s) as for the evaluation of the dual channel architecture.

Table 4 shows the system unreliability for different levels of truncation. Truncating at the third level of failure yields a system unreliability of $1.17 \cdot 10^{-4}$. This is a slight improvement with respect to the system unreliability yielded by the pure-redundancy architecture, which was $5.20 \cdot 10^{-4}$. However, although

the improvement is not very significant in terms of system unreliability, it will be shown later in this analysis that the most significant improvement is the removal of several single points of failure.

Table 4
Enhanced dual channel architecture: single points of failure and unreliability for different levels of truncation and an evaluation time of $500h$.

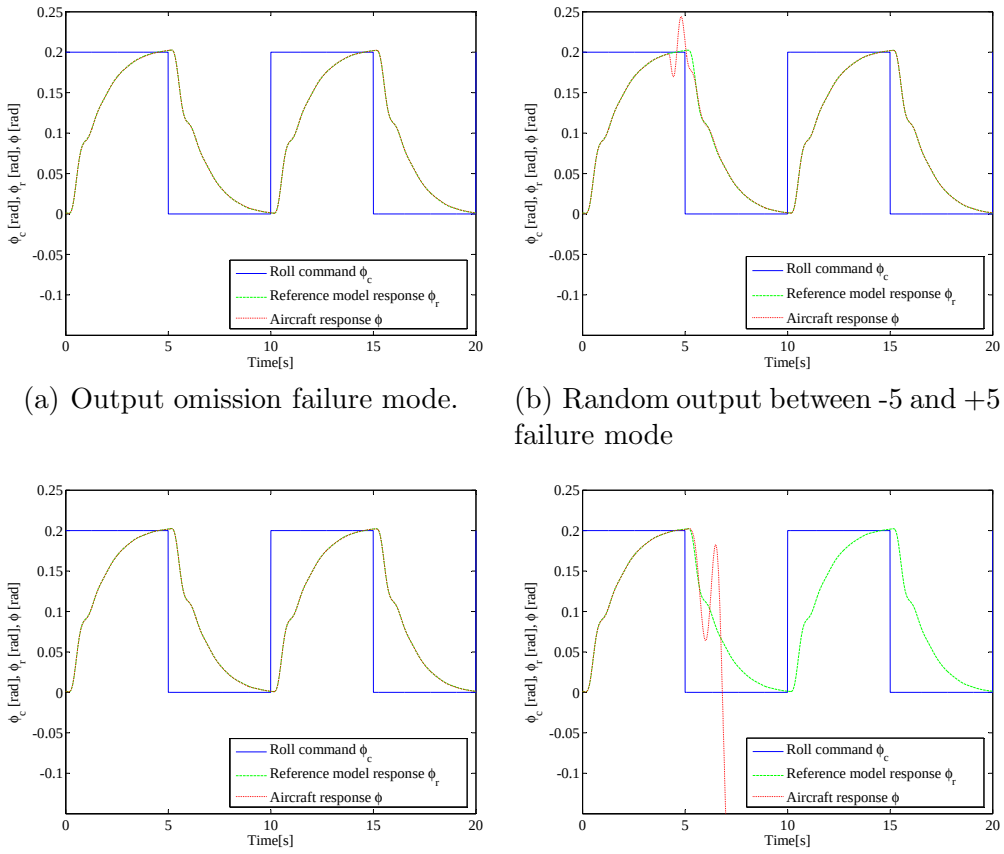
Truncation level	*Unreliability lower bound*	*Unreliability upper bound*	Single points of failure	*# of system configurations*
2	$1.14 \cdot 10^{-4}$	$1.87 \cdot 10^{-4}$	5	68
3	$1.17 \cdot 10^{-4}$	$1.17 \cdot 10^{-4}$	5	3924

Single primary flight computer failures

With the FDIR mechanism, it can be seen that except the failure-by-delayed-output, Fig. 9(d), which cause the system to fail; the other PFCs failures are detected and isolated, and the aircraft stays stable after the control laws in the remaining PFC are reconfigured, Fig. 9(a) — Fig. 9(c). Additionally, the aircraft behavior is almost unaffected for a failure-by-output-omission, Fig. 9(a); and for a failure-by-stuck-output, Fig. 9(c). For a failure-by-random-output, there is a small transient after the failure occurs, as it can be seen in Fig. 9(b), but the aircraft recovers in less than 2s.

Primary flight computer failures after failures in the PFC-SD

Although not shown here, single failures by-omission, or by-commission in a PFC-SD do not affect the aircraft performance at all. In the case of a single failure-by commission of a PFC-SD, the computer which has the PFC-SD will be shut down despite the fact that the computer did not fail itself, and the control laws in the remaining computer will be reconfigured. Although these failures won't cause the system to fail, if not announced, they can create a dangerous situation in which the system is believed that both computers are operational, when they are not. A first failure-by-omission in a self-detection circuit followed by a failure in its own PFC will go undetected causing the system to fail. The effects on the aircraft dynamic behavior for this situation are similar to those for the Dual channel architecture after any first failure in



(a) Output omission failure mode.

(b) Random output between -5 and +5 failure mode

(c) Output stuck failure mode.

(d) Output delayed failure mode.

Fig. 9. Enhanced dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in one of the primary flight computers, and a failure injection time $t_f = 4$ s.

one of the computers, displayed in Fig. G.6.

To announce the failure-by-commission of a self-detection circuit is an easy task, since its effect is the same as shutting down a computer due to any other failure. The failure-by-omission is trickier, unless there is a self-checking routine for the self-detection circuit as well.

5.3 Dual-Dual Channel Architecture

The FDIR mechanism introduced in the previous section improves the system overall performance in that the unreliability is smaller and several single failures have been removed. However, it is not perfect since it is not capable

of detecting failure-by-delayed output in the PFCs. In this section, an improved FDIR mechanism will be introduced, which, hopefully, will handle all the failures within the PFCs.

The proposed further improved architecture, called dual-dual channel architecture (DDCa) is, in this case, very similar to the EDCa architecture. The main architectural difference is that rather than having just one main processor within each PFC with a failure self-detection circuit; now, each PFC will implement a pair of lock-step processors receiving the same inputs and doing exactly the same computations. Additionally a dual-comparator circuit PFC-DC within the primary flight computer will compare their outputs. As in the EDCa design, each PFC exchanges information of its status (failed or operational) with the other PFC. The control laws within the processors of each PFC are reconfigurable depending on the status of the other PFC. With these additional elements the FDIR implemented has the following features:

- **Detection.** The dual-comparator circuit within each PFC will check whether the outputs of the processor-pair agree or not. If the outputs disagree, the dual-comparator reports a failure.
- **Isolation.** Once the dual-comparator circuit detects a failure, both lock-step processors are shut down.
- **Reconfiguration.** Once the dual-comparator circuit detects a failure, a reconfiguration signal is sent to the remaining PFC to double the gain of the control surface controllers on each lock-step processor.

Table 5
Dual-dual channel architecture: primary flight computers' dual-comparator circuits failure model parameters.

Component	*Failure modes*	*Description*	U_b	$\lambda(/h)$
PFC_1-DC, PFC_2-DC	Omission	Output set to zero regardless the input	1	10^{-8}
	Commission	Output set to one regardless the input	2	10^{-8}

Table 5 displays the failure model information of the dual-comparator circuits within each PFC. The failure models parameters for the rest of the component is the same as for the dual channel architecture, Table 1.

The use of lock-step processors requires the introduction of an additional processor on each PFC, thus increasing the complexity and the cost as well. However, the use of lock-step processors should result in a perfect detection, isolation, and reconfiguration of any failure within a PFC. Thus, removing all the PFC-related single points of system failure.

Table 6
Dual-dual channel architecture: single points of failure and unreliability for different levels of truncation and an evaluation time of $500h$.

Truncation level	*Unreliability lower bound*	*Unreliability upper bound*	Single points of failure	*# of system configurations*
2	$1.49 \cdot 10^{-5}$	$9.47 \cdot 10^{-5}$	3	76
3	$1.51 \cdot 10^{-5}$	$1.51 \cdot 10^{-5}$	3	4640

Performance and Reliability Evaluation

The improved architecture was evaluated under the same conditions as the pure-redundancy architecture and the enhanced dual channel architecture, see Section 5.1 for the details.

Table 6 shows the system unreliability estimates. Truncating at the third level of failure yields a system unreliability of $1.51 \cdot 10^{-5}$ and a truncation error of $2.76 \cdot 10^{-7}$. This is an improvement of one order of magnitude with respect to the results of both the dual architecture and the enhanced dual architecture.

Figure G.10 completes the analysis of the dual-dual architecture. It shows the aircraft behavior for a failure-by-delayed output within one of the processors of a PFC. The effect of other failures in one of the processors within the PFC has a similar effect. As it can be seen, the aircraft performance is not altered at all by any of these failures, thus the FDIR provided by the lock-step processors scheme is utterly satisfactory. Although not shown here, single failures in the PFC-DC cause similar problems to those mentioned in the analysis of the enhanced dual channel architecture for single failures in any PFC-SD. Thus, a failure-by-omission or by-commission in a PFC-DC will not affect the system performance at all for similar reasons to those discussed in the the last paragraph of section 5.2. Similarly to that case, for first failure-

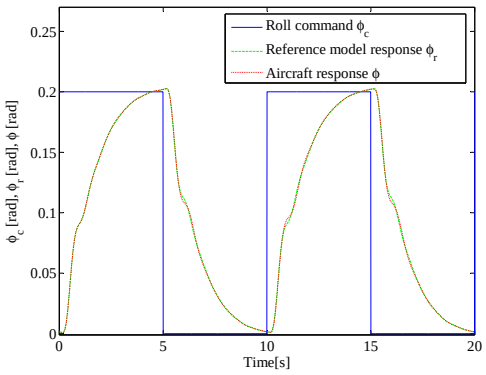


Fig. 10. Dual-dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for an output delayed failure modes in one of the processors of a primary flight computer, and a failure injection time $t_f = 4$ s.

by-omission in a dual comparator circuit followed by a failure in its own PFC will go undetected causing the system to fail.

Table 7
Results comparison.

Architecture	*Unreliability*	*PFC-originated single points of failure*	*Control-surface-originated single points of failure*
Dual channel	$5.20 \cdot 10^{-4}$	8	3
Enhanced dual channel	$1.17 \cdot 10^{-4}$	2	3
Dual-dual channel	$1.51 \cdot 10^{-5}$	0	3

5.4 Architectures Comparison

Table 7 summarizes the main analysis results for the three architectural solutions. The unreliability of both the DCa and the EDCa designs is very similar; however, the EDCa design removed 75% of the single points of system failure due to PFC failures with respect to the DCa design. The introduction of the lock-step processors in the DDCa design made a huge impact on the results. The unreliability is one order of magnitude smaller than in the other two designs, and all the single points of system failure due to PFC failures were removed. However, it is important to note that the DCa design is the less complex one in terms of operation, i.e., no need for detecting, isolating or reconfiguring the system; and in the number of components. These two issues makes this architecture less expensive to produce and to maintain. On

the other hand, the DDCa design is much more complex than the other two: more components, and more complex operation, which makes this design more expensive to produce and to maintain. However, the impact on reliability and fault-tolerance is very important. Therefore, it is the work of the designer to make trade-offs between complexity, and cost; and performance, reliability, and fault-tolerance.

6 Concluding Remarks

The methodology presented in this paper uses a quantitative mathematical model of the behavior of the system to be analyzed. The model includes not only the system nominal behavior (no component failures), but also degraded behaviors due to failed components. This is an important feature of the methodology, since the system evaluation in the presence of failures no longer relies on the judgment of the analyst to assess whether or not a sequence of component failures will cause the system to fail or not. Furthermore, by using a quantitative system behavioral model, it is possible to evaluate degraded system operational modes, i.e., the analysis is not system fails or not, the analysis now allows us to assess the "degree of failure" of a system. The other advantage of the present methodology is that it can help the system designer to uncover weak points in the design and quantify the influence of the different system performance and reliability drivers.

The analysis shown in this paper opens up further research questions. The design improvement shown here was done manually, analyzing the results for each iteration and using expert judgment to guide the changes in the design. Guiding the design in a structured and automatic way is a challenging problem worth exploring, i.e., how to extend the methodology to uncover, in a structured fashion, the problems that most impact a design in terms of performance and reliability, and how they could be addressed.

Acknowledgment

The authors would like to thank Dr. Philip S. Babcock at the Charles Stark Draper Laboratory for his invaluable ideas and fruitful discussions, and for his constant support of this research. Dr. Piero Miotto at the Charles Stark Draper Laboratory provided the IMU model and fruitful discussions. Thanks also to Thomas T. Myers at Systems Technology, Inc. for providing technical information regarding the linear lateral-directional aircraft dynamics model.

A Primary Flight Computer

A.1 Voter

$$\epsilon_1 = |u_1 - u_2|$$
$$\epsilon_2 = |u_1 - u_3|$$
$$\epsilon_3 = |u_2 - u_3|$$

$$\tilde{u} = \begin{cases} \sum_{i=1}^{3} u_i - \max_i\{u_i\} - \min_i\{u_i\}, & \text{if } \exists\ i,j\ /\ \epsilon_i + \epsilon_j < 2\epsilon \\ \frac{u_i + u_j}{2}, & \text{if } \exists\ i\ /\ \epsilon_i < \epsilon \\ NIL, & otherwise \end{cases} \tag{A.1}$$

where $\epsilon = 0.1$rad.

A.2 Control Laws

A.2.1 Roll Control

$$R_r(s) = K_{r_1}\phi_c(s) + K_{r_2}R_b(s) + K_{r_3}\frac{s + z_r}{s + p_r}P_b(s)$$
$$\delta_{a_r^*}^{l(r)}(s) = (P_r + \frac{I_r}{s} + D_r s)(R_r(s) \pm K_r\delta_a^{l(r)}(s))$$

$$\delta_{a_r}^{l(r)}(t) = \begin{cases} \delta_{a_r^*}^{l(r)}(t), & \text{for } r = 0 \\ 2\delta_{a_r^*}^{l(r)}(t), & \text{for } r = 1 \end{cases}$$

$$\hat{\delta}_{a_r}^{l(r)}(t) = \begin{cases} \delta_{a_r}^{l(r)}(t), & \text{for } U_B = 0 \\ 0, & \text{for } U_B = 1 \\ rand(l,u), & \text{for } U_B = 2 \\ \delta_{a_r}^{l(r)}(\tau), & \text{for } U_B = 3 \\ \sigma_{\tau_l}(\delta_{a_r}^{l(r)}(t)), & \text{for } U_B = 4 \end{cases} \tag{A.2}$$

where $K_{r_1} = 0.66$, $K_{r_2} = -0.145\text{s}$, $K_{r_3} = 2.16\text{s}$, $z_r = 11.1\text{s}^{-1}$, $p_r = 25\text{s}^{-1}$, $P_r = 0.45\text{A}$, $I_r = 6\text{A/s}$, $D_r = 0.01\text{As}$, and $K_r = -1.33$ were taken from [35]; and $l = -5\text{A}$, $u = 5\text{A}$ $\tau_l = 0.2\text{s}$, and τ is the time at which the computer gets stuck.

A.2.2 Yaw Control

$$R_y(s) = K_{y_1}\delta_c(s) + \frac{s^2 + z_{y_1}s + z_{y_2}}{s(s+p_y)}(K_{y_2}R_b(s) + K_{y_3}P_b(s))$$
$$\delta_{r_r^*}(s) = (P_y + \frac{I_y}{s} + D_y s)(R_y(s) + K_y\delta_r(s))$$

$$\delta_{r_r}(t) = \begin{cases} \delta_{r_r^*}(t), & \text{for } r = 0 \\ 2\delta_{r_r^*}(t), & \text{for } r = 1 \end{cases}$$

$$\hat{\delta}_{r_r}(t) = \begin{cases} \delta_{r_r}(t), & \text{for } U_B = 0 \\ 0, & \text{for } U_B = 1 \\ rand(l,u), & \text{for } U_B = 2 \\ \delta_{r_r}(\tau), & \text{for } U_B = 3 \\ \sigma_{\tau_l}(\delta_{r_r}(t)), & \text{for } U_B = 4 \end{cases} \tag{A.3}$$

where $K_{y_1} = 0.001$, $K_{y_2} = -0.7816\text{s}$, $K_{y_3} = 0.172\text{s}$, $z_{y_1} = -0.00125\text{s}^{-2}$, $z_{y_2} = -0.001875\text{s}^{-1}$, $p_y = 1.5\text{s}^{-1}$, $P_y = 0.45\text{A}$, $I_y = 6\text{A/s}$, $D_y = 0.01\text{As}$, $K_y = -1.33$, were taken from [35]; and $l = -5\text{A}$, $u = 5\text{A}$ $\tau_l = 0.2\text{s}$, and τ is the time at which the computer gets stuck.

B Inertial Measurement Unit

$$x_{imu}(t+\Delta t) = x_{imu}(t) + randn(0, \sigma_{ss}\sqrt{1-e^{-2\Delta t/\tau}})$$
$$x_{imu}(0) = \sigma_{ss}$$

$$y_{imu}(t+\Delta t) = x_{imu}(t+\Delta t) + [M_1 + M_2 + I]\begin{bmatrix} p_b \\ q_b \\ r_b \end{bmatrix} + \begin{bmatrix} randn(\mu_b, \sigma_b) \\ randn(\mu_b, \sigma_b) \\ randn(\mu_b, \sigma_b) \end{bmatrix}$$

$$M_1 = K_1 \begin{bmatrix} randn(0,1) & 0 & 0 \\ 0 & randn(0,1) & 0 \\ 0 & 0 & randn(0,1) \end{bmatrix}$$

$$M_2 = K_2 \begin{bmatrix} 0 & 0 & 0 \\ -randn(0,1) & 0 & 0 \\ randn(0,1) & 0 & -randn(0,1) \end{bmatrix}$$

$$\hat{y}_{imu} = \begin{cases} y_{imu}, & \text{for } U_B = 0 \\ 0, & \text{for } U_B = 1 \\ Gy_{imu}, & \text{for } U_B = 2 \\ y_{imu} + B, & \text{for } U_B = 3 \end{cases} \tag{B.1}$$

where $\sigma_{ss} = 3.15 \cdot 10^{-6}$, $\Delta t = 0.01$s, $\tau = 100$s, $\mu_b = 4.85 \cdot 10^{-6} rand(0,1)$, $\sigma_b = 2.09 \cdot 10^{-5}$, $K_1 = 10^{-4}$, $K_2 = 9.7 \cdot 10^{-5}$, $G = 1.5$, and $B = 0.3$deg.

C Actuation Subsystem

C.0.3 Ailerons

$$G(s) = k_1 \frac{(s+\frac{1}{\tau_c})(s+\frac{1}{\tau_m})}{(s+\frac{1}{\tau_1})(s+\frac{1}{\tau_2})(s+\frac{1}{\tau_3})}$$
$$T^{l(r)}_{a_{1(2)}}(s) = k_2 \frac{G(s)}{1+G(s)} \hat{\delta}^{l(r)}_{a_r}(s)$$

$$\hat{T}^{l(r)}_{a_{1(2)}}(t) = \begin{cases} T^{l(r)}_{a_{1(2)}}(t), & \text{for } U_B = 0 \\ 0, & \text{for } U_B = 1 \\ T^{l(r)}_{a_{1(2)}}(\tau), & \text{for } U_B = 2 \end{cases} \tag{C.1}$$

where $k_1 = 5000\text{s}^2$, $k_2 = 50$, $\tau_c = 10$s, $\tau_m = 10$s, $\tau_1 = 4$s, $\tau_2 = 10$s, $\tau_3 = 100$s, and τ is the time at which the actuation subsystem gets stuck.

C.0.4 Rudder

$$G(s) = k_1 \frac{(s+\frac{1}{\tau_c})(s+\frac{1}{\tau_m})}{(s+\frac{1}{\tau_1})(s+\frac{1}{\tau_2})(s+\frac{1}{\tau_3})}$$
$$T_{r_{1(2)}}(s) = k_2 \frac{G(s)}{1+G(s)} \hat{\delta}_r(s)$$

$$\hat{T}_{r_{1(2)}}(t) = \begin{cases} T_{r_{1(2)}}, & \text{for } U_B = 0 \\ 0, & \text{for } U_B = 1 \\ T_{r_{1(2)}}(\tau), & \text{for } U_B = 2 \end{cases} \tag{C.2}$$

where $k_1 = 5000\text{s}^2$, $k_2 = 50$, $\tau_c = 10\text{s}$, $\tau_m = 10\text{s}$, $\tau_1 = 4\text{s}$, $\tau_2 = 10\text{s}$, $\tau_3 = 100\text{s}$, and τ is the time at which the actuation subsystem gets stuck.

D Mechanical Combiner

D.1 Ailerons

$$T_a^{l,r} = G_a(\hat{T}_{a_1}^{l(r)} + \hat{T}_{a_2}^{l(r)}) \tag{D.1}$$

where $G_a = 0.5$.

D.2 Rudder

$$T_r = G_r(\hat{T}_{r_1} + \hat{T}_{r_2}) \tag{D.2}$$

where $G_r = 0.5$.

E Control Surfaces

E.1 Ailerons

$$\dot{x}_a^{l,r} = -\frac{1}{\tau_a} x_a^{l,r} + \frac{1}{\tau_a} T_a^{l,r}$$
$$\delta_a^{l(r)} = \sigma_{\tau_l}(x_a)$$

$$\hat{\delta}_a^{l(r)} = \begin{cases} \delta_a^{l(r)}, & \text{for } U_B = 0 \\ \delta_a(\tau), & \text{for } U_B = 1 \\ \alpha_0, & \text{for } U_B = 2 \end{cases} \tag{E.1}$$

where $\tau_a = 0.04$s and $\tau_l = 0.11$s, $\alpha_0 = 0.216$rad, were taken from [35]. τ is the time at which the aileron gets stuck.

E.2 Rudder

$$\begin{aligned} \dot{x}_r &= -\frac{1}{\tau_r} x_r + \frac{1}{\tau_r} T_r \\ \delta_r &= \sigma_{\tau_l}(x_r) \end{aligned}$$

$$\hat{\delta}_r = \begin{cases} \delta_r, & \text{for } U_B = 0 \\ \delta_r(\tau), & \text{for } U_B = 1 \\ \Psi, & \text{for } U_B = 2 \end{cases} \tag{E.2}$$

where $\tau_r = 0.05$s and $\tau_l = 0.1$s were taken from [35]. τ is the time at which the rudder gets stuck.

F Control surfaces position sensor

F.1 Ailerons

$$\begin{aligned} \dot{x}_{s_a}^{l(r)} &= -\frac{1}{\tau_s} x_{s_a}^{l(r)} + \frac{1}{\tau_s} \delta_a^{l(r)} \\ y_{s_a}^{l(r)} &= \frac{\lceil \sigma_{\tau_l}(x_{s_a}^{l(r)}) \frac{1}{R} \rceil}{\frac{1}{R}} \end{aligned}$$

$$\hat{y}_{s_a}^{l(r)} = \begin{cases} y_{s_a}^{l(r)}, & \text{for } U_B = 0 \\ 0, & \text{for } U_f = 1 \\ G y_{s_a}^{l(r)}, & \text{for } U_B = 2 \\ y_{s_a}^{l(r)} + B, & \text{for } U_B = 3 \end{cases} \tag{F.1}$$

where $\tau_s = 0.01$s, $\tau_l = 0.001$s, $R = 0.001$, $G = 1.5$, and $B = 0.3$deg.

F.2 Rudder

$$\begin{aligned} \dot{x}_{s_r} &= -\frac{1}{\tau_s} x_{s_r} + \frac{1}{\tau_s} \delta_r \\ y_{s_r} &= \frac{\lceil \sigma_{\tau_l}(x_{s_r}) \frac{1}{R} \rceil}{\frac{1}{R}} \end{aligned}$$

$$\hat{y}_{s_r} = \begin{cases} y_{s_r}, & \text{for } U_B = 0 \\ 0, & \text{for } U_B = 1 \\ G y_{s_r}, & \text{for } U_B = 2 \\ y_{s_r} + B, & \text{for } U_B = 3 \end{cases} \tag{F.2}$$

where $\tau_s = 0.01$s, $\tau_l = 0.001$s, $R = 0.001$, $G = 1.5$, and $B = 0.3$deg.

G Linear lateral-directional aircraft dynamics

$$\begin{bmatrix} \dot{\beta} \\ \dot{p_b} \\ \dot{r_b} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{Y_\beta}{V} & \sin\alpha_0 & -\cos\alpha_0 & \frac{g\cos\alpha_0}{V} \\ L_\beta & L_p & L_r & 0 \\ N_\beta & N_p & N_r & 0 \\ 0 & 1 & \tan\alpha_0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ p_b \\ r_b \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 0 & Y_{\delta_r} \\ N_{\delta_a^l} & N_{\delta_a^r} & N_{\delta_r} \\ L_{\delta_a^l} & L_{\delta_a^r} & L_{\delta_r} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a^l \\ \delta_a^r \\ \delta_r \end{bmatrix} \tag{G.1}$$

where $Y_\beta = -50.69 m/s^2$, $V = 178 m/s$, $\alpha_0 = 0.216 rad$, $g = 9.81 m/s^2$, $L_\beta = -32.3 s^{-2}$, $L_p = -0.374 s^{-1}$ $L_r = 2.40 s^{-1}$, $N_\beta = 1.06 s^{-2}$, $N_p = -0.0406 s^{-1}$, $N_r = -0.0809 s^{-1}$, $Y_{\delta_r} = 0.0179 s^{-1}$, $N_{\delta_a^l} = -3.175 s^{-2}$, $N_{\delta_a^r} = 3.175 s^{-2}$, $N_{\delta_r} = 6.66 s^{-2}$, $L_{\delta_a^l} = -0.855 s^{-2}$, $L_{\delta_a^r} = 0.855 s^{-2}$, and $L_{\delta_r} = -1.18 s^{-2}$, were taken from [41].

Notation

A :	Markov model state-transition matrix
B_s :	Bias factor for sensor behavioral model example
c_i :	Component i
$c_j, r_j^{i,k}$:	Parameters for the requirements of the j^{th} performance metric
$DPRA$:	Dynamic probabilistic risk assessment
DCa :	Dual channel architecture
$EDCa$:	Enhanced dual channel architecture
$ESCS$:	Event sequences and consequences spectrum
$DDCa$:	Dual-dual channel architecture
$FDIR$:	Failure detection, isolation, and recovery
$f_{c_i}^j(\cdot,\cdot)$:	State evolution function for the j^{th} behavioral mode of component c_i
$f_s^{i,k}(\cdot,\cdot)$:	State evolution function for system configuration $\{i,k\}$
$g_{c_i}^j(\cdot,\cdot)$:	Output function for the j^{th} behavioral mode of component c_i
$g_s^{i,k}(\cdot,\cdot)$:	Output function for system configuration $\{i,k\}$
$G_s, \hat{G}_s$:	Gain for sensor behavioral model example
$h_j(\cdot)$:	Reward model function for the j^{th}performance metric
IMU :	Inertial measurement unit
LA :	Left aileron
$LAAS$:	Left aileron actuation subsystem
$LAPS$:	Left aileron position sensor
PCA :	Propulsion controlled aircraft

PFC :	Primary flight computer
$PFC-SD$:	Primary flight computer failure self-detection circuit
$PFC-DC$:	Primary flight computer dual-comparator circuit
p_b :	Roll rate
p_{b_r} :	Reference model roll rate
$p_{i,k}(t)$:	Probability that, at time $t \geq 0$, the system configuration $\{i,k\}$ is declared as non-failed, given that at $t=0$ is in configuration $\{1,0\}$
$p^*_{i,k}(t)$:	Probability that, at time $t \geq 0$, the system configuration $\{i,k\}$ is declared as failed, given that at $t=0$ is in configuration $\{1,0\}$
$P(t)$:	System configurations probability vector
Q :	System unreliability
Ru :	Rudder
R :	System reliability
RAS :	Rudder actuation subsystem
RPS :	Rudder position sensor
RA :	Right aileron
$RAAS$:	Right aileron actuation subsystem
$RAPS$:	Right aileron position sensor
r_b :	Yaw rate
r_{b_r} :	Reference model yaw rate
r :	Reward function
r_S :	Reward function associated to the reliability measure computation
$r_{\overline{S}}$:	Reward function associated to the unreliability measure
$r_{\overline{P}}$:	Reward function for the power aggregated performance measure example
$r_\beta, r_{p_b}, r_{r_b}, r_\phi$:	Case-study performance metrics parameters
r_{Z_j} :	Reward model associated with the j^{th} performance metric
$s_{i,k}$:	Indicator function for the status of configuration $\{i,k\}$
R_s :	Resolution for the sensor behavioral model example
T :	System global evaluation time
t :	Time
t_f :	Failure injection time
U_B :	Behavioral modes random variable
$u_s(t)$:	System input
$u_{c_1}(t)$:	Input for sensor behavioral model example
$u_{c_i}(t)$:	Component c_i input
$U_{c_i}(t)$:	Component c_i behavioral modes random variable
V :	Forward velocity
$x_{c_1}(t)$:	State variable for sensor behavioral model example
$x_{c_i}(t)$:	Component c_i state variables
$x_s(t)$:	System state variables
$X(t)$:	Random variable associated to the system configuration at time t
$y_s(t)$:	System output
$y_{c_1}(t)$:	Output for sensor behavioral model example
$y_{c_i}(t)$:	Component c_i output
$Y(t)$:	Random variable associated to the status of a Markov chain at time t
Z_j :	j^thPerformance metric
α_0 :	Pitch angle
β :	Sideslip angle
β_r :	Reference model sideslip angle
δ_a^l, δ_a^r :	Left and right aileron angles
δ_r :	Rudder angle
λ :	Failure rate
λ_{lm} :	l-to-m component behavioral mode transition rate
λ_{NO} :	Nominal-to-omission behavioral mode transition rate for the sensor example
λ_{NG} :	Nominal-to-gain-change behavioral mode transition rate for the sensor example
λ_{NB} :	Nominal-to-bias behavioral mode transition rate for the sensor example
λ_{GO} :	Gain-change-to-omission behavioral mode transition rate for the sensor example
λ_{BO} :	Bias-to-omission behavioral mode transition rate for the sensor example
$\lambda^{i,k}_{j,k-1}$:	Transition rate that causes the system to go from configuration $\{j,k-1\}$ to $\{i,k\}$
ϕ :	Roll angle
ϕ_c :	Roll command
ϕ_r :	Reference model roll angle
$\Phi^{i,k}_{Z_j}$:	Requirements for the j^{th} performance metric when the system is in configuration $\{i,k\}$
$\sigma_{\tau_l}(\cdot)$:	τ_l-shift operator
τ_l :	Latency for sensor behavioral model example
τ_s :	Inverse of bandwidth for sensor behavioral model example
$\lceil\cdot\rceil$:	Ceiling function

$\mathbb{E}[\cdot]$: Expectation

References

[1] E. Gai, M. Adams, Measures of merit for fault-tolerant systems, Tech. Rep. CSDL-P-1752, The Charles Stark Draper Laboratory, Cambridge, MA (1983).

[2] A. Avižienis, Design of fault-tolerant computers, in: Proceedings of the Fall Joint Computer Conference, AFIPS Conference, Washington, DC, 1967, pp. 733–743.

[3] E. Moore, C. Shannon, Reliable circuits using less reliable relays, parts I & II, Journal of the Franklin Institute 262 (1956) 191–208 and 281–297.

[4] H. Watson, Launch control system safety study, Bell Laboratories Technical Report 1.

[5] J. Dugan, S. Bavuso, M. Boyd, Dynamic fault fault-tree models for fault-tolerant computer systems, IEEE Transactions on Reliability 41 (3) (1992) 363–377.

[6] P. Babcock, An introduction to reliability modeling of fault-tolerant systems, Tech. Rep. CSDL-R-1899, The Charles Stark Draper Laboratory, Cambridge, MA (1987).

[7] A. Høyland, M. Rausand, System Reliability Theory, John Wiley and Sons, New York, NY, 1994.

[8] A. Reibman, M. Veeraraghavan, Reliability modeling: An overview for system designers, IEEE Computer 24 (4) (1991) 49–57.

[9] M. Malhotra, K. Trivedi, Power-hierarchy of dependability model types, IEEE Transactions on Reliability 43 (3) (1994) 493–502.

[10] A. Dominguez-Garcia, J. Kassakian, J. Schindall, Reliability evaluation of the power supply of an electrical power net for safety-relevant applications, Journal of Reliability Engineering and System Safety 91 (5) (2006) 505–514.

[11] N. Leveson, Safeware: System Safety and Computers, Addison-Wesley Publishing Company, Boston, MA, 1995.

[12] J. Taylor, An algorithm for fault-tree construction, IEEE Transactions on Reliability 31 (2) (1982) 219–254.

[13] Y. Papadopoulos, D. Parker, C. Grante, Model-based automated synthesis of fault trees from matlab-simulink models, in: Proceedings of the International Conference on Dependable Systems and Networks, Gothenburg, Sweden, 2001, pp. 77–82.

[14] J. Taylor, An integrated approach to the treatment of design and specification errors in eletronic systems and software, in: Proceedings of the Fifth European Conference on Electrotechnics, Copenhagen, Denmark, 1982.

[15] Y. Papadopoulos, D. Parker, M. Walker, U. Pertersen, R. Hamann, Q. Wu, Automated failure modes and effects analysis of systems on-board ship, in: Proceedings of the International Conference on Marine Research and Transportation, Ischia, Italy, 2005.

[16] A. Amendola, Event sequences and consequence spectrum: A methodology for probabilistic transient analysis, Nuclear Science An Engineering 77 (3) (1981) 297–315.

[17] T. Aldemir, Computer-assisted markov failure modeling of process control systems, IEEE Transactions on Reliability R-36 (1) (1987) 133–144.

[18] J. Devooght, C. Smidts, Probabilistic reactor dynamics-I: The theory of continuos event trees, Nuclear Science and Engineering 111 (a) (1992) 229–240.

[19] G. Grimmett, D. Stirzaker, Probability and Random Processes, 3rd Edition, Oxford University Press, Oxford, UK, 2001.

[20] J. Devooght, C. Smidts, Probabilistic dynamics as a tool for dynamic psa, Reliability Engineering and System Safety 52 (3) (1996) 185–196.

[21] C. Smidts, J. Devooght, Probabilistic reactor dynamics-II: A monte carlo study of a fast reactor trasient, Nuclear Science and Engineering 111 (a) (1992) 241–256.

[22] P. Labeau, C. Smidts, S. Swaminathan, Dynamic reliability: Towards an integrated platform for probabilistic risk assessment, Journal of Reliability Engineering and System Safety 68 (3) (2000) 219–254.

[23] Y. Hu, A guided simluation methodology for dynamic probabilistic risk assessment of complex systems, Ph.D. thesis, University of Maryland, College Park, MD (2005).

[24] J. Meyer, Computation-based reliability analysis, IEEE Transactions on Computers C-25 (6) (1976) 578–584.

[25] M. Beaudry, Performance-related reliability measures for computing systems, IEEE Transactions on Computers C-27 (6) (1978) 548–560.

[26] J. Meyer, On evaluating the performability of degradable computing system, IEEE Transactions on Computers C-29 (8) (1980) 720–731.

[27] R. Sahner, K. Trivedi, A. Puliafito, Performance and Reliability Analysis of Computer Systems, Kluwer Academic Publishers, Boston, MA, 1996.

[28] D. Luenberger (Ed.), Introduction to Dynamic Systems, John Wiley, New York, NY, 1979.

[29] K. Trivedi, M. Malhotra, R. Fricks, Markov reward approach to performability and reliability analysis, in: Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Durham, NC, 1994, pp. 7–11.

[30] A. Reibman, Modeling the effect of reliability on performance, IEEE Transactions on Reliability 39 (3) (1990) 314–320.

[31] D. Liberzon, Switching in Systems and Control, Birkhauser, Boston, MA, 2003.

[32] A. Domínguez-García, An integrated methodology for the performance and reliability evaluation of fault-tolerant systems, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA (2007).

[33] F. Cristian, Understanding fault-tolerant distributed systems, Communications of the ACM 34 (2) (1991) 56–78.

[34] A. Domínguez-García, J. Kassakian, J. Schindall, J. Zinchuk, On the use of behavioral models for the integrated performance and reliability evaluation of fault-tolerant avionics systems, in: Proceedings of DASC 2006, Portland, OR, 2005.

[35] D. McRuer, T. Myers, P. Thompson, Literal singular-value-based flight control system design techniques, AIAA Journal of Guidance 12 (6) (1989) 913–919.

[36] D. McRuer, I. Ashekenas, D. Graham, Aircraft Dynamics and Automatic Control, Princeton University Press, Princeton, NJ, 1973.

[37] J. Roskan, Flight Dynamics of Rigid and Elastic Airplanes, The University of Kansas, Lawrence, KS, 1972.

[38] J. Dugan, M. Veeraraghavan, M. Boyd, N. Mittal, Proceedings of the eighth symposium on reliable distributed systems, in: Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium, Seattle, WA, 1989.

[39] P. Babcock, G. Rosch, J. Zinchuk, An automated environment for optimizing fault-tolerant systems design, in: Proceedings of the Reliability and Maintainability Symposium, Orlando, FL, 1991, pp. 360–367.

[40] T. Tucker, Touchdown: the development of propulsion controlled aircrafts at nasa dreyden, Monograph in aerospace history no. 16, NASA (1999).

[41] D. McRuer, T. Myers, Advanced piloted aircraft flight control system design methodology volume I: Knowledge base, Contractor Report 181726, NASA, Langley, VA (1988).

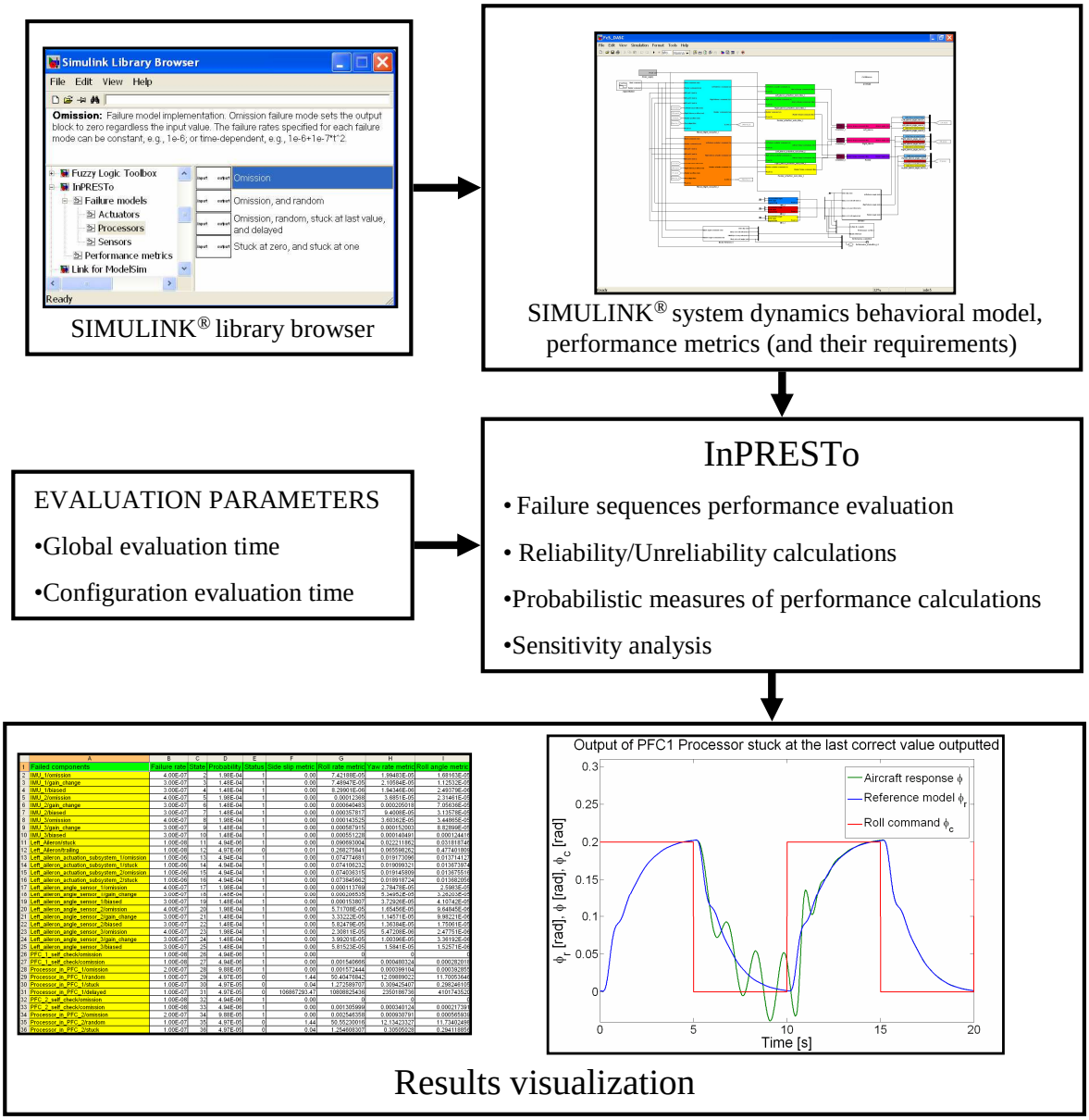
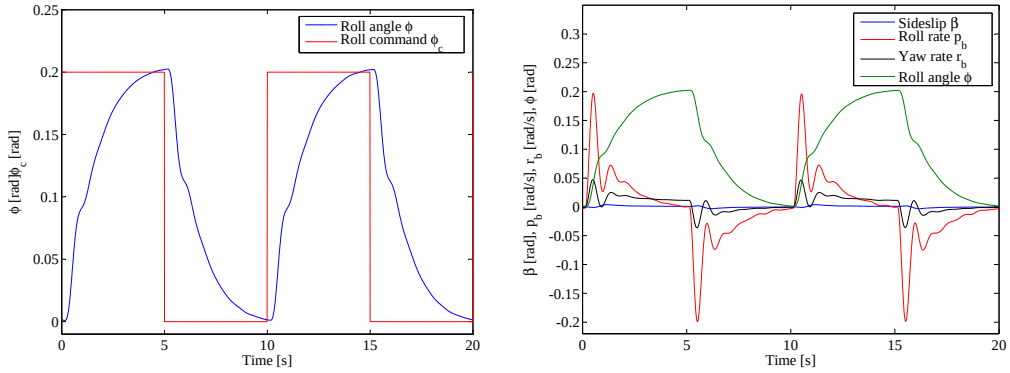


Fig. G.1. Basic functionality of InPRESTo.



(a) Roll command ϕ_c, and corresponding roll angle response ϕ.

(b) Sideslip response β, roll rate response p_b, yaw rate response r_b, and roll angle response ϕ.

Fig. G.2. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in the roll command ϕ_c.

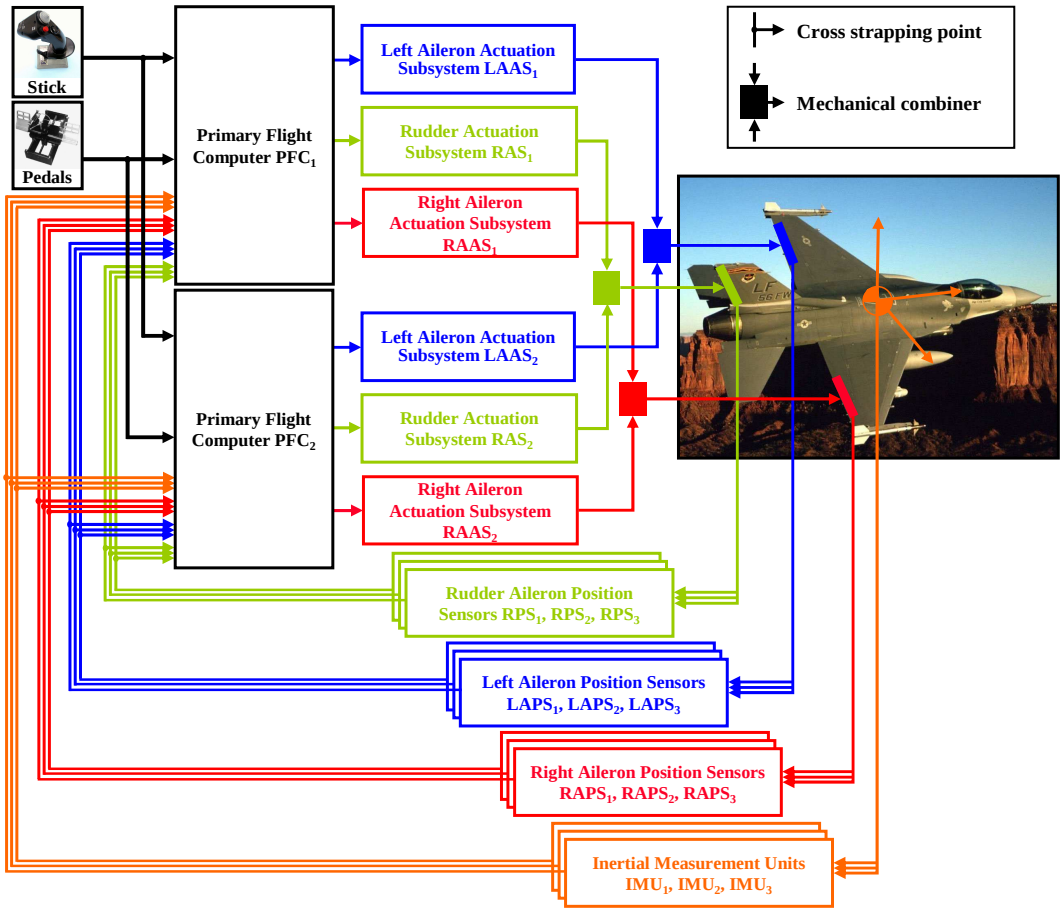
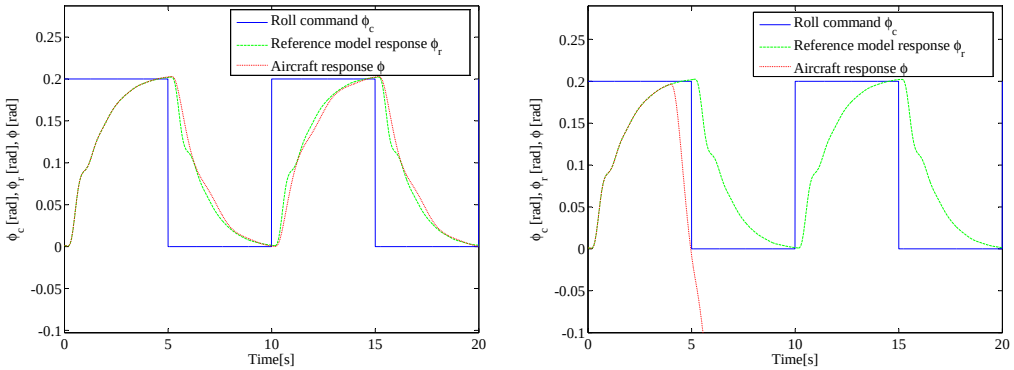


Fig. G.3. Lateral-Directional Flight Control System Fault-Tolerant Architecture.



(a) Stuck failure mode. (b) Trailing failure mode.

Fig. G.4. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in the left (or right) aileron, and a failure injection time $t_f = 4$ s.

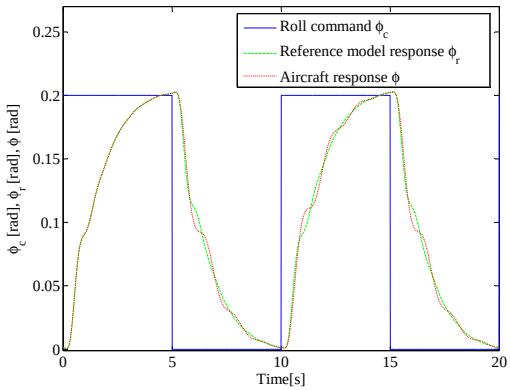
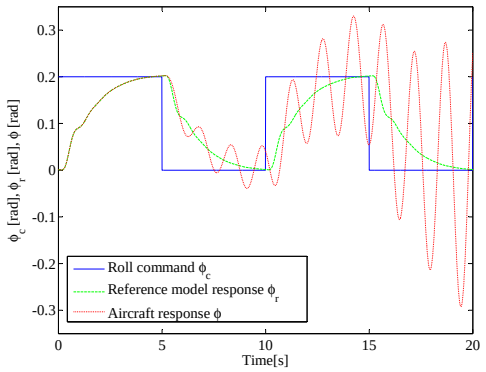
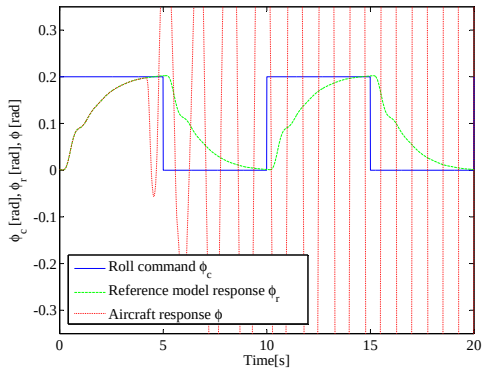


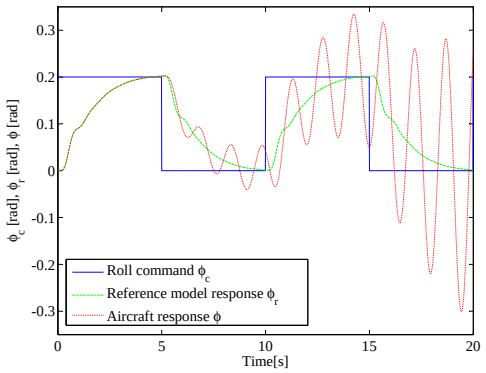
Fig. G.5. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for a failure-by-stuck in one of the left (or right) aileron actuation subsystems, and a failure injection time $t_f = 4$ s.



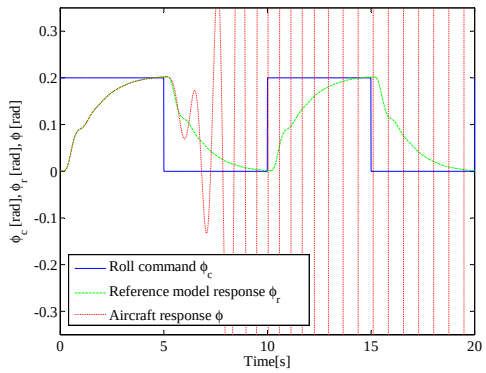
(a) Output omission failure mode.



(b) Random output between -5 and +5 failure mode

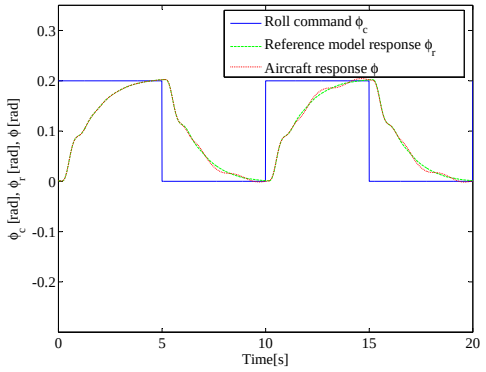


(c) Output stuck failure mode.

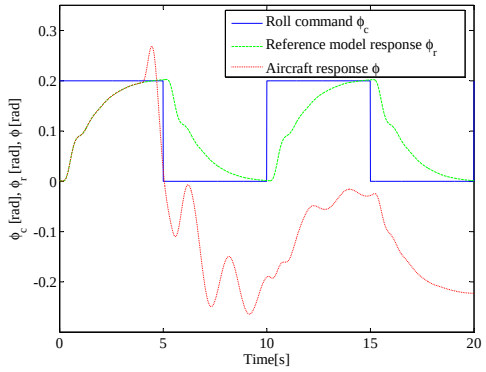


(d) Output delayed failure mode.

Fig. G.6. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in one of the primary flight computers, and a failure injection time $t_f = 4$ s.



(a) Stuck failure mode.



(b) Trailing failure mode.

Fig. G.7. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in the rudder, and a failure injection time $t_f = 4$ s.

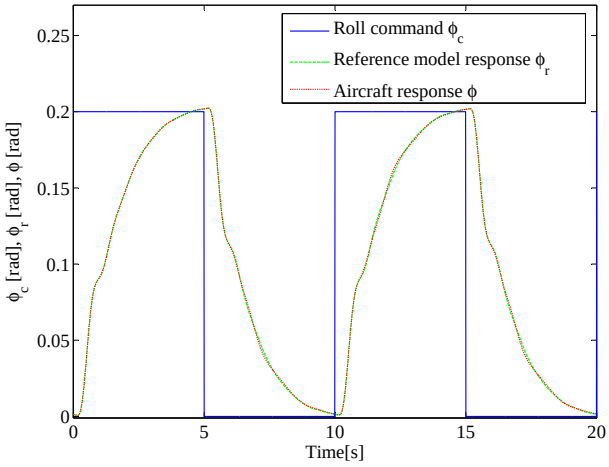
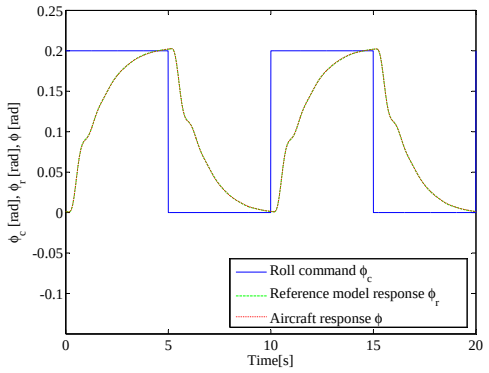
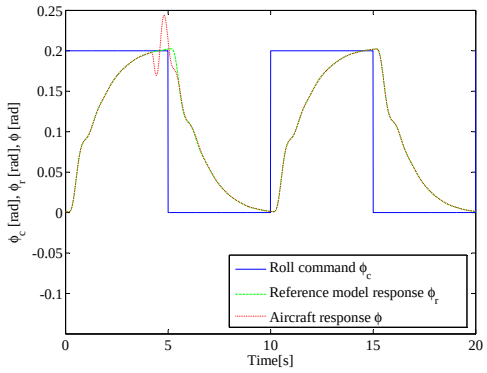


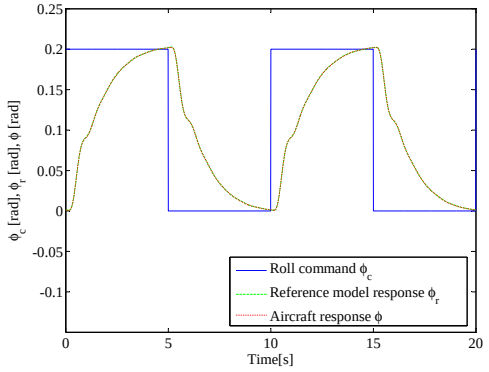
Fig. G.8. Dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for a failure-by-stuck in one of the rudder actuation subsystems, and a failure injection time $t_f = 4$ s.



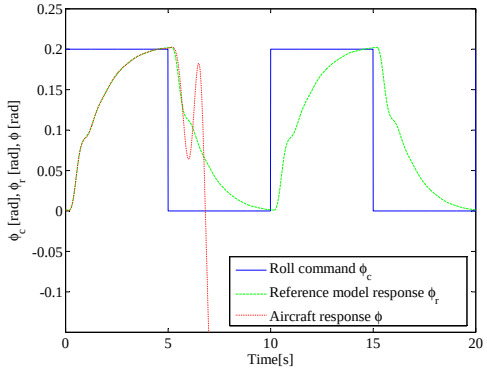
(a) Output omission failure mode.



(b) Random output between -5 and +5 failure mode



(c) Output stuck failure mode.



(d) Output delayed failure mode.

Fig. G.9. Enhanced dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for different single failure modes in one of the primary flight computers, and a failure injection time $t_f = 4$ s.

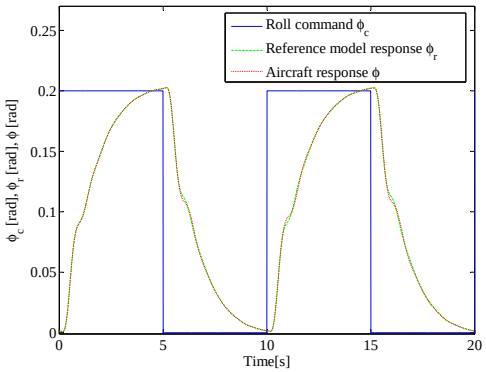


Fig. G.10. Dual-dual channel architecture performance in response to a 0.2rad, 0.1Hz square wave in roll command ϕ_c. Roll angle aircraft response ϕ compared to reference model response ϕ_r, for an output delayed failure modes in one of the processors of a primary flight computer, and a failure injection time $t_f = 4$ s.