Analytically Redundant Controllers for Fault Tolerance: Implementation with Separation of Concerns

Kashif Hameed, Rob Williams, Jim Smith

Abstract

Diversity or redundancy based software fault tolerance encompasses the development of application domain specific variants and error detection mechanisms. In this regard, this paper presents an analytical design strategy to develop the variants for a fault tolerant real-time control system. This work also presents a generalized error detection mechanism based on the stability performance of a designed controller using the Lyupanov Stability Criterion. The diverse redundant fault tolerance is implemented with an aspect oriented compiler to separate and thus reduce this additional complexity. A Mathematical Model of an Inverted Pendulum System has been used as a case study to demonstrate the proposed design framework.

Keywords: diversity and redundancy, analytically redundant controllers, aspect oriented design and programming, fault tolerance, fault injection.

1. Introduction

The application of redundancy and diversity in order to tolerate hardware, software and environmental faults is not new [1]. The use of redundancy with respect to time, design and data has been proved successful in improving the dependability of computer based systems. Since exact duplication of software components cannot increase reliability when forced with software design faults, some sort of diversity in the design and implementation is required. The phrase "design differently fails differently" summarizes the situation. So, a group of diverse redundant variants communicates only dependable output. The decision about the trusted output is made with the help of adjudicators, voters or an acceptance test. Dependability may be further enhanced by distributing these variants on different hardware platforms.

Recovery Blocks (RcB) scheme is one of the two original diverse software fault tolerance techniques. It was introduced by Horning, et al. [2], with early implementations developed by Randell [3] and Hetcht [4]. It utilizes sequential execution of software variants and trusty output is communicated via an acceptance test. It is a dynamic design diverse technique and output of a variant is communicated as soon as the acceptance test is passed otherwise next variant is tested. The hardware fault-tolerant architecture equivalent to the RcB scheme is stand-by sparing or passive dynamic redundancy.

N-Version Programming (NVP) is also one of the original diverse software fault tolerance design techniques suggested by Elmendorf [5] and developed by Avizienis and Chen [6, 7]. It is static technique in which a task is executed by several processes or programs sequentially on a single computer, or concurrently on different computers and a result is accepted only if it is adjudicated as an acceptable result, through a voting mechanism. It is a static design diverse technique as all the variants are executed and a final voted result is communicated. The hardware fault tolerance architecture related to the NVP is N-modular redundancy.

The hybrid of these two design diverse techniques form the basis of various other design diverse techniques like Distributed Recovery Blocks (DRB) Kim [8] and Consensus Recovery Blocks (CRB) suggested by Scott [9].

Although a number of schemes to improve software dependability of computer bases systems are proposed, there is a need to address low level issues like designing variants and generalized error detection mechanism for particular application domains. Most real time control software serves mission critical or safety critical systems. It is therefore required to address some cost effective design strategy that enhance the dependability. The coming sections address analytically redundant controllers design strategy in this regard. These redundant controllers are backed with a model based error detection mechanism. The proposed scheme is demonstrated with an inverted pendulum model based case study. In order to reduce the implementation complexity of proposed scheme, an aspect oriented version of such dependable framework is also provided. This framework is ported to Matlab/Simulink using an S-Function Wrapper for the validation of proposed scheme.

2. Analytically Redundant Controllers Design

The development of redundant software components or variants often requires some performance criterion on the basis of which they are developed and ranked. A very common example often provided in the academia

are the sorting algorithms [1] that are designed and ranked based on their size and execution times.

In the domain of real-time control systems, performance and stability measures such as settling time overshoot and rise time may be used to design and rank controller variants. Thus the variants designed and developed based on such analytical measures are said to be analytically redundant. One such definition is provided by Lui [10] for real time controllers. In his definition attributes like reliability, performance and stability have been chosen as analytical measures. According to him a software component C1 is analytical redundant to C2 with respect to Q_j, the measure of quality attribute provided:

$\mathbf{R_i} < \mathbf{Q_j}(C1) \le \mathbf{Q_j}(\mathbf{C}2)$, Where $\mathbf{R_i}$ is the minimal requirement of Q_j

For example, controllers are said analytically redundant with respect to maintaining stability of the physical system in a feasible region bounded by system states and physical constraints if each one of the controllers asymptotically stabilizes the physical system inside the given region.

In order to demonstrate and analyze the design strategy of analytically redundant controllers, an inverted pendulum system model has been chosen.

2.1. Inverted Pendulum System

The inverted pendulum (IP) system is a very common example used to validate a variety of controller design strategies because of its non-linear dynamics and inherently unstable behavior.

The IP system consists of a cart, driven by a DC motor, and a pendulum attached to the cart. The cart can move along a horizontal track, and the pendulum is able to rotate freely in the range of [-30^o, 30^o] with respect to vertical in the vertical plane parallel to the track. There is no direct control applied to the pendulum. The position of the cart $\mathbf{x}$ and angle θ are measured through two potentiometers. The dynamics of the system are described by the state of the system, which consists of the cart position $\mathbf{x}$, the cart velocity $\dot{\mathbf{x}}$, the pendulum angle θ, and the pendulum angular velocity $\dot{\theta}$.

The physical system has state and control constraints. Specifically, the cart position is restricted in a range [-0.7, 0.7] meters, the maximum speed of the cart is 1.0 meter/second, the angle is constrained to [-30^o, 30^o] and the motor input voltage is limited in the range [-4.96, 4.96] volts.

A linearized state space model of IP system is derived to following set of matrices as shown below.

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \qquad \ldots\ldots (1)$$
$$\mathbf{y} = \mathbf{Cx}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{\mathbf{B_x}}{\overline{\mathbf{M}} - \dfrac{\mathbf{mL}}{\overline{\mathbf{L}}}} & -\dfrac{\mathbf{gmL}}{\overline{\mathbf{L}}\left(\overline{\mathbf{M}} - \dfrac{\mathbf{mL}}{\overline{\mathbf{L}}}\right)} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{\mathbf{B_x}}{\overline{\mathbf{M}}\left(\overline{\mathbf{L}} - \dfrac{\mathbf{mL}}{\overline{\mathbf{M}}}\right)} & \dfrac{\mathbf{g}}{\overline{\mathbf{L}} - \dfrac{\mathbf{mL}}{\overline{\mathbf{M}}}} & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ \dfrac{\mathbf{K_m}}{\overline{\mathbf{M}} - \dfrac{\mathbf{mL}}{\overline{\mathbf{L}}}} \\ 0 \\ -\dfrac{\mathbf{K_m}}{\overline{\mathbf{M}}\left(\overline{\mathbf{L}} - \dfrac{\mathbf{mL}}{\overline{\mathbf{M}}}\right)} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{x_c} & \dot{\mathbf{x}}_\mathbf{c} & \theta & \dot{\theta} \end{bmatrix}^\mathbf{T}$$

$$\overline{\mathbf{M}} = \mathbf{m} + \mathbf{M}, \ \overline{\mathbf{L}} = \frac{\mathbf{I} + \mathbf{mL}^2}{\mathbf{mL}}$$

Parameter	Value
M (mass of cart)	0.5 Kg
m (mass of pendulum	0.2 Kg
L (Half length of pendulum)	0.3 m
B_x (Viscous Friction Co-efficient)	0.1 Kg/s
I (moment of inertia)	0.006 Kg-m^2
g (Acceleration due to gravity)	9.8 m/s^2
K_m(Input to Force Gain)	1

Analytically Redundant IP Controller:

A number of control synthesis are possible for inverted pendulum system stabilization counting from non-robust conventional PID [11] and state-space based pole placement [12] to more robust Optimal Control Strategies [13]. In this work we will concentrate on optimal state feedback control using linear quadratic regulator (LQR) [14].

In a LQR design, the feedback gain matrix K for a linear state feedback control law $\mathbf{u} = -\mathbf{Kx}$ is found by minimizing a quadratic cost function of the form:

$$\mathbf{J} = \int_0^\infty \mathbf{x}^\mathbf{T}(\mathbf{t})\mathbf{Qx}(\mathbf{t}) + \mathbf{u}^\mathbf{T}(\mathbf{t})\mathbf{Ru}(\mathbf{t})\mathbf{dt}$$

Q and R are weighting parameters that penalize certain states $\mathbf{x}$ and control inputs $\mathbf{u}$ respectively. By varying the matrix Q and Scalar R, the analytically redundant controller gains are obtained and the resulting control algorithms may asymptotically stabilize the system at the operating condition X=0.

The linearized model of IP system as in equation (1) has been used to design and two different analytically redundant controllers by choosing two different R: R = 0.01 and R=0.1 and assigning equal weigh age to penalty on states

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LQR problem has been solved to obtain the following set of analytically redundant feedback gains:

$\mathbf{K1} = [-10.0000 \ -15.8365 \ -79.1980 \ -18.5048], \mathbf{R1} = 0.01$

$\mathbf{K2} = [-3.1623 \ -5.3136 \ -33.4676 \ -7.1619], \mathbf{R2} = 0.1$

These two variants of IP controller are simulated to compare the performance measures of these variants.

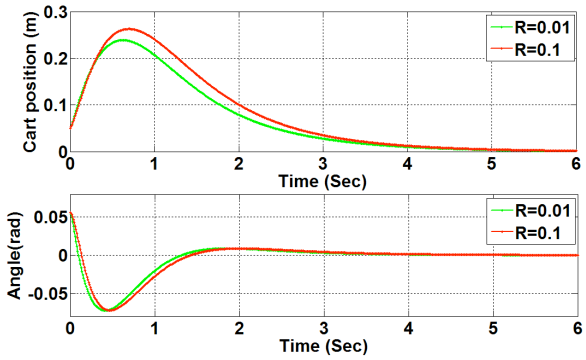


Figure 1 K1 K2 Simulation Response

The performance of the two variants is measured with following parameters as tabulated below

Table 1 Performance parameters

Performance Parameters	K1 (R=0.01)	K2 (R=0.1)
Settling time	5.41	5.57
Overshoot	0	0
Maximum derivation	0.2396	0.2636
Settling time on quadratic state error (QSE)	2.46	2.7
Steady-state value of accumulated QSE	0.6687	1.2506

It can be seen that controller K1 performs better than K2 although both the controllers asymptotically stabilize the inverted pendulum to the equilibrium position. The next section investigates and compares safety and stability characteristics of analytically redundant controllers and deduces a design principle based on that.

2.2. Lyapunov Stability based Error Detection

Every dynamical system has constraints due to physical limits, external environment and operating conditions. These absolute behavior constraints also dictate the boundary of a safe operating region. For example the inverted pendulum case study under consideration has constraints on cart displacement, car speed, pendulum angular movement and maximum voltage of the DC motor. These state constraints are represented by a polygon in an n-dimensional space as shown in figure 10. A trusty inverted pendulum controller must generate a subset of the states within the safety region and also ensure that future trajectories must also lie within these bounds. If so the stability and safety of the system is ensured. This problem has been formalized mathematically using Lyapunov stability criterion [15]. The solution of Lyapunov criterion based problem provides a Lyapunov function V(x) illustrated by an ellipsoid in Fig. 10. The breaching of safety region boundary indicates a faulty system state or controller. This indication is used to signal an exception or switchover a redundant controller for recovery.

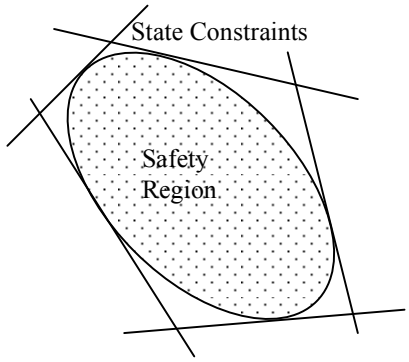


Figure 2 Safety Region and State Constraints

The IP controller problem is solved with LMI solver tool CVX [16] to attain the safety regions (Lyapunov Function) for three variants of IP controller. In addition to the two controllers already designed as LQR problem discussed earlier, the third controller is designed such that it has largest safety region (SR). An important task is to formulate the problem before presenting to CVX tool.

Problem Formulation

We are considering a quadratic Lyapunov function $\mathbf{V(x)} = \mathbf{x^T P x}$ where $\mathbf{P}$ is a positive definite matrix for a Linear Time Invariant (LTI) system defined as $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$. The necessary condition for Lyapunov stability criterion $\dot{\mathbf{V}}(\mathbf{x}) \le 0$ can be easily derived to a linear matrix inequality (LMI) $\mathbf{A^T P} + \mathbf{PA} < 0$.

It is worth noting that $\mathbf{V(x)}$ is not unique for a given system and controller combination, it is therefore required to find largest safety region such that state constraints polytope may not be restricted. The maximum volume of ellipsoid defined by the safety region $\mathbf{S} = \{\mathbf{x} : \mathbf{x^T P x} \le 1\}$ is equivalent to minimize (log det P) [17]. The final thing is the LMI based formulation of input, output constraints of the system.

The complete set of problem is formulated and presented to CVX tool as follows:

find Q that
minimizes logdetQ^{-1} /*Maximise Ellipsoidal Volume*/
subject to

$\mathbf{Q > 0}$ /* Positive Definite and Symmetric*/

$\mathbf{Q\bar{A}^T + \bar{A}Q < 0}$ /*LMI based Stability Criterion*/

$\alpha_i^T \mathbf{Q} \alpha_i \le 1, \mathbf{i} = 1...\mathbf{p}$ /*Input State Constraints*/

$\beta_j^T \mathbf{Q} \beta_j \le 1, \mathbf{j} = 1...\mathbf{r}$ /*Output Constraints*/

Next we consider a case where a controller with largest safety region is to be found. Thus in this case K is also unknown and we need to find K such that $P=Q^{-1}$ presents the largest safety region. Let $Z=KQ^{-1}$, the complete LMI problem now becomes:

find Q and Z that
minimizes logdetQ^{-1} /*Maximise Ellipsoidal Volume*/
subject to

$\boldsymbol{Q > 0}$ /*Positive Definite and Symmetric*/

$\mathbf{QA^T + AQ + Z^TB^T + BZ < 0}$ /*LMI Stability Criterion*/

$\alpha_i^T \mathbf{Q} \alpha_i \le 1, \mathbf{i} = 1...\mathbf{p}$ /*Input State Constraints*/

$$\begin{bmatrix} 1 & \beta_j^T \mathbf{Z} \\ \mathbf{Z}^T \beta_j & \mathbf{Q} \end{bmatrix} \ge 0, \mathbf{j} = 1...\mathbf{r}$$ /*Output Constraints*/

The inverted pendulum model has following state and output constraints,

$\alpha_1^T = [5 \quad 0 \quad 0 \quad 0]$ $\alpha_2^T = [-5 \quad 0 \quad 0 \quad 0]$

$\alpha_3^T = [0 \quad 1 \quad 0 \quad 0]$ $\alpha_4^T = [0 \quad -1 \quad 0 \quad 0]$

$\alpha_5^T = [0 \quad 0 \quad 3.82 \quad 0]$ $\alpha_6^T = [0 \quad 0 \quad -3.82 \quad 0]$

$\beta_1^T = \frac{\mathbf{K}}{4.95}$ $\beta_2^T = \frac{-\mathbf{K}}{4.95}$

The solution of above two cases using CVX tool provides $\mathbf{P = Q^{-1}}$ and thus safety region for known controller gains. Moreover the gain matrix $\mathbf{K = ZQ^{-1}}$ thus obtained presents the largest safety region. The resulting safety regions have been projected to x1~x2 plane (x3=x4=0) and x3~x4 plane (x1=x2=0) as shown below:

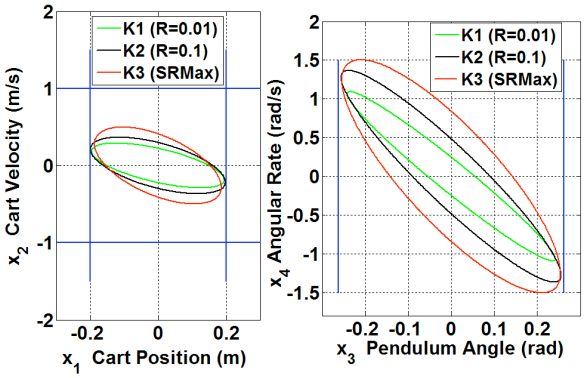
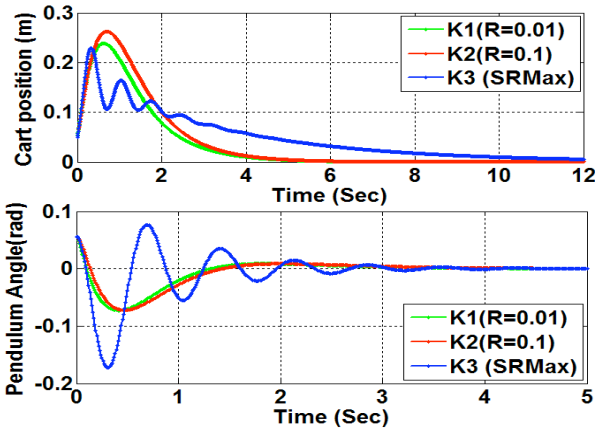


Figure 3 Safety Regions Comparison

In order to propose a design principle for analytically redundant controllers designed above, a performance comparison by the help of a simulation has been done.



Finally table 2 & 3 summarizes the comparison of performance and safety measures of the inverted pendulum case study.

Table 2 Cart Position Performance

Performance Parameters	K1	K2	K3
Settling time (seconds)	5.41	5.57	14.61
Overshoot (meters)	0	0	0
Maximum derivation (meters)	0.2396	0.2636	0.23
Measure of size of safety region ($\sqrt{\det Q}$)	0.0074	0.0172	0.043

Table 3 Pendulum angle Performance

Performance Parameters	K1	K2	K3
Settling time (seconds)	3.13	3.39	3.62
Overshoot (meters)	0.0726	0.0723	0.173
Maximum derivation (meters)	0.0726	0.2636	0.173
Measure of size of safety region ($\sqrt{\det Q}$)	0.0074	0.0172	0.043

It can be observed that controllers K1 and K2 show better performance characteristics like improved settling time, less overshoot but less stability/safety region. The controller K3 on the other hand presents a larger stability/safety region but poor performance.

The performance and the safety region analysis of the three variants reveal that high performance controllers exhibit smaller safety regions and vice versa. Thus high performance controllers are selected as primary controllers and safety controller as secondary one. The three variants for IP controller are termed as

experimental, base-line and safety controller for future reference. We are using the boundary of ellipsoid associated with safety controller as error detection and switching rule.

The above debate proposes an analytically redundant design framework applied to an inverted pendulum system. Moreover Lyapunov stability criterion based error detection mechanism has also been presented. However an important consideration is dispatching a dependable output to the external environment (plant). It is required to switch to appropriate controller as desired. In this regard we present a state machine that acts a switching logic upon faulty controller detection.

2.3. State Machine Based Switching Logic

The IP controller may be in one of the three states {experimental, base-line, safety} ranked from high performance to large safety enveloped controllers respectively. Once a controller is marked faulty, it is disabled and we need two boolean state variables base controller ready (*bc_ready*) and experimental controller ready (*ec_ready*) to keep track of which controller is available. In order to describe the behavior of the physical system with relation to system safety and recovery from a faulty situation, we define boolean variables *safe* and *to_bc* with the following assignments:

- If physical system is safe, *safe*=1 else *safe*=0
- If the active controller is safety controller and system is ready for base-line control *to_bc*=1 else *to_bc*=0

The state transition of the active controller is determined by boolean variables *bc_reay, ec_ready, safe, to_bc*. Fig. 11 shows the state transition diagram of the active controller when the boolean expressions on the transition arcs evaluated to true

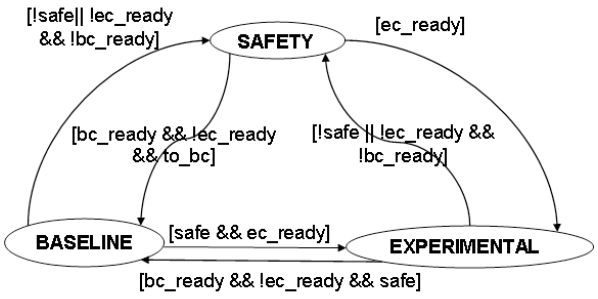


Figure 4 State Transition for Active Controller Selection.

3. Separation of Fault Tolerance Concerns

Unfortunately fault tolerant diverse redundant mechanisms do not come for free, bringing additional cost and complexity to the core application. J. Xu [18] has argued that application developers now have to address both application dependent and redundancy-related concerns in an intrusive way that complicates the task of implementing and maintaining fault-tolerant software.

One of solutions to reduce the additional complexity is by separating and modularizing these non-functional concerns from the core functionality. The evolving area of Aspect-Oriented Programming & Design (AOP&D) supports the modularized implementation of crosscutting concerns. We are benefitting from the power of aspect oriented compiler AspectC++ [19] to weave redundancy related concerns in non-fault tolerant experimental IP controller. The variants of IP controller, error detection mechanism and state machine based switching logic are separated in a well modularized aspect. The static advice declarations like slice() of AspectC++ are used to extend the IP controller object functionality by weaving these concerns. Moreover dynamic advice declarations like before(), after() and around() are used to execute the fault tolerant strategy using a fault tolerant framework and communicating the non-faulty output to the external plant.

4. AOFT Controller Implementation in Simulink

A number of safety and mission critical control applications are developed: designed and simulated in modeling/simulation based environments like Matlab/Simulink [20]. Rather than conducting fault injection experiments in the real world environment, with all the associated cost and potential damage to equipment and life, it is easy to employ software models of the physical real time systems in which the software is to be integrated or employed. The simulation environments also help in designing and conducting a variety of fault injection experiments.

In order to validate the proposed design strategy, the AOFT IP controller is ported to modeling/simulation environment Matlab/Simulink using C++ S-Function Wrapper [21]. The transformation process involves the mapping of the output file to an S-function C++ Wrapper after the weaver has merged the functional and aspect code. The IP system is also modeled in Simulink.

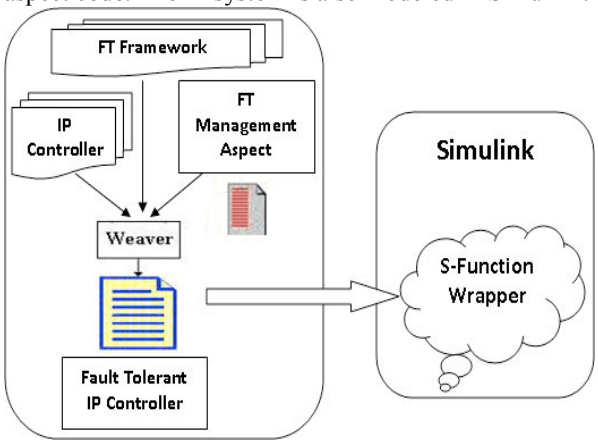


Figure 5 AOFT Controller Porting to S-Function

The transformation steps are detailed as follows:

- The weaved code with inline and wrapper functions for advice and aspects against the join points is ported to global memory space in simulation structure file template.
- Appropriate simulation work pointers are assigned to the FT Object
- The modified AspectC++ wrapper functions are called through Simulink work pointers.
- Appropriate Input & Output ports are assigned to interact with the physical world model.

5. Fault Injection based Controller validation

Fault injection is a deliberate introduction of faults to access the dependability of computer based systems or validate a fault tolerant strategy. Faults may be injected at the interface between components and external environment or within a component itself. One of commonly used technique is software implemented fault injection (SWIFI). In this technique faults are injected at the actual software of a computing system by corrupting code or data [22]. Code Mutation is also a specific form of static fault injection where source code is changed instead of program/systems state. Actually different versions of the same program are created by making small syntactic changes [23].
The effect of actuator faults on an actuator control has been modeled and simulated in a study by Theilliol et al. [24]. They model the faults as additive and multiplicative effects on the output delivered by the controller. $\mathbf{u}_j^f = \alpha \mathbf{u}_j + \mathbf{u}_{j0}$

Where $\mathbf{u}_j^f$ represents the faulty jth output, $0 \le \alpha \le 1$ is a gain degradation factor, $\mathbf{u}_{j0}$ is the constant offset in the control output. Moreover sensor and actuator faults have also been addressed as additive and multiplicative bias at the input and output in separate studies by Kerrigan and Theilliol et al. [25, 26].

The validation of our proposed design strategy is also done by injecting realizable faults. In order to provide better test coverage, faults are injected in the input/output interfaces between controller and rest of environment and also within the controller component. The faults considered are ill designed faulty gains of the controller and wrong polarity emulating a programming fault. The input/output faults are considered additive and multiplicative in nature simulating common sensor and actuator fault models.

Faults are injected in the primary (experimental controller) whereas other variants (base-line and safety controllers) are assumed fault free.

5.1. Programming & Design Faults in IP Controller

In this case a mutant IP controller having inherent faulty gains or wrong or opposite signs is used.

Case1: Faulty Controller Gains

The mission objective is to move the cart from its initial position to 10 cm the left of the centre (-0.1m) such that the inverted pendulum remains vertically stable. The gains of the primary controller (experimental controller) are tampered to faulty (ill designed) values.

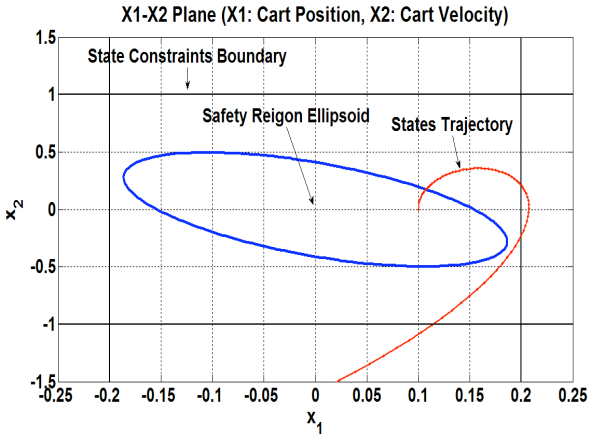


Figure 6 Safety Region & Trajectory without AOFT

Without AOFT

It can be observed from fig. 13 that IP system is unsafe and unstable as the system states cross the safety envelope and then boundary constraints as well. Thus the mission objective is not achieved and results in a failure.

With AOFT

It can be seen that from fig. 14 that upon detection of bug at 0.26sec (error detection latency), the active controller is switched to the safety control, and remained under safety control until the physical system is ready for the baseline control even the system is safe at 0.37sec. The Lyapunov function based check monitors whether the system is ready for base-line controller. Thus at 0.67sec the system is switched to intermediate performance base line controller and remained in control afterwards.

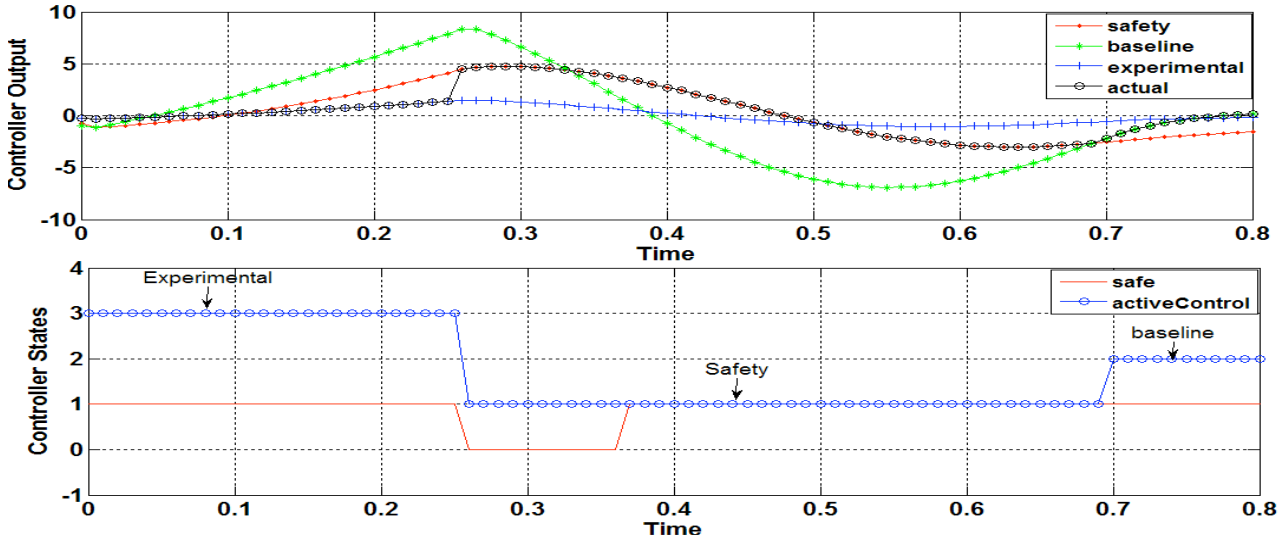


Figure 7 (a) Analytical Redundant Controllers Output (b) Safe: 1-Safe 0-Unsafe, Active Control

Figure 15 presents the trajectories of the physical system and safety region plot. The trajectory terminates within the safety region thus demonstrates that the system is asymptotically stable and also converges to the commanded values.

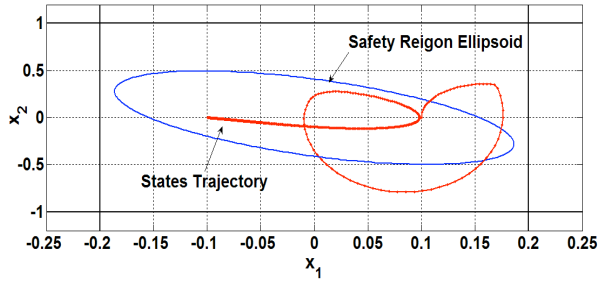


Figure 8 Safety Region & Trajectory with AOFT

Case2: Sign Bugs

In this case a mutant IP controller having wrong or opposite signs is used.

Summary: The IP system is unstable and unsafe without AOFT support as shown in figure 16. However in the presence of AOFT, the fault is detected by the help of safety check and recovered by switching over to safety and base line controller for safe operation as shown in figure 16.

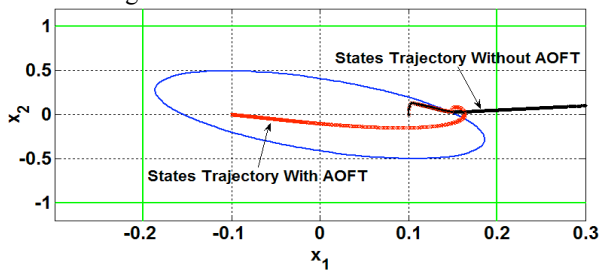


Figure 9 Safety Region & Trajectory

5.2. Input & Output Interface Faults in IP Controller

In this case input/output interface faults are simulated by injecting multiplicative-additive faults ($u^f=\alpha u+u_0$), where u represents non-faulty input or output, α is gain degradation factor and u_0 is an additive bias.

Case1: Multiplicative Output Faults

Here we did not consider the additive bias thus $u_0=0$ and only α is varied from 10% to 90% of actual output. It is observed that up to 30% gain degradation may results in 43% of steady state error that is undetected and recovery check is not validated with this degraded performance.

Next we are considering some cases where partial and total failures are avoided by introducing proposed AOFT framework.

Partial Failure Avoided (SS Error of 0.1m)

In this case output of experimental controller is degraded by a factor of 0.5 started at about 2sec and AOFT is enabled at 5sec. it is observed that 100% steady state (SS) error is introduced without AOFT. As soon as the AOFT is enabled, it results in system recovery from this erroneous state and improves the performance characteristics in terms of steady state error. A steady state error may be considered as a partial failure that is avoided in this case.

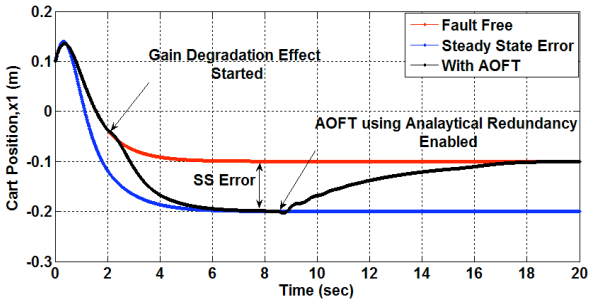


Figure 10 Recovery from Partial Failure

Total Failure Avoided

In this case output of experimental controller is degraded by a gain degradation factor of 0.1 started at about 2sec. it is observed that without fault tolerance measures the system is unsafe and unstable. However, the proposed methodology avoided this failure. As soon as the fault is detected, safety controller takes over first and later delegates the control to base line controller for better performance as shown below.

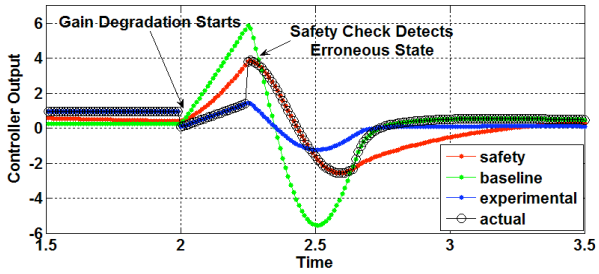


Figure 11 Active Controller Output

Case2: Additive Output Faults

Here we did not consider the multiplicative effect thus α=1. It has been observed that offset (u_0) added to the output resulted in steady state error and makes system unstable and unsafe. Again the proposed methodology prevents such failures.

Simulating External Push Pull Force

Next we consider the case where a sinusoidal fault distribution is applied and then remove after some time. It simulates a transient external push pull force applied by a human or some faulty output interface mechanism. The result of this is the oscillation build up that may lead to a failure. The proposed safety check provides a basis to predict such occurrence before the system is not controllable as shown below. The proposed AOFT helps recovering the system from this failure.

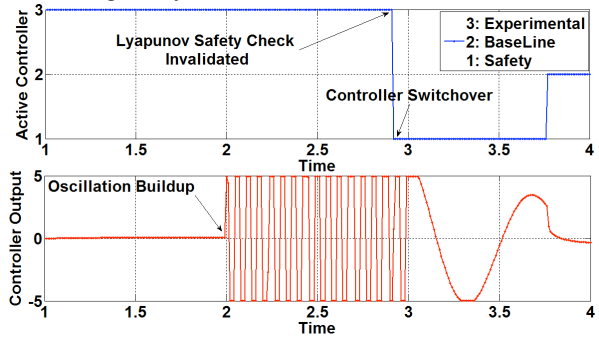


Figure 12 Sinusoidal Fault & Recovery

Case3: Additive Input Faults

In this case additive faults are induced at the input stage of the controller either simulating noisy input from the sensors or buggy software interface between sensor and controller. A sinusoidal disturbance is added to the input stage of experimental controller. The proposed strategy tolerates such faults by switching to analytical redundant variants.

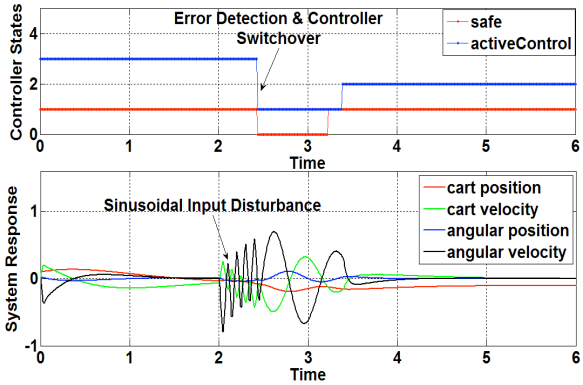


Figure 13 Sinusoidal Input Disturbance

6. Conclusions & Limitations

The work presented above adds on to the existing knowledge in fault tolerance by proposing a design strategy that dictates how diversity or redundancy based software fault tolerance frameworks may be incorporated in safety or mission critical control components. It is demonstrated that fault tolerance management issues like definition, initialization and execution can be handled at meta-level i.e. in a separate aspect. This would keep the functional design and code oblivious to fault tolerance concerns. Moreover this would help decreasing the tangling of functional code with fault tolerant concerns as well.

This work presents an analytically redundant design framework for incorporating diverse redundant software fault tolerance strategies. The performance and safety characteristic of three analytically redundant controllers for inverted pendulum system are compared and deduces that high performance controllers possess less safety margin and vice versa. Thus the controller agility compromises the safety or stability.

The aspect oriented fault tolerant inverted pendulum controller has been ported to Modeling Simulation environment Matlab/Simulink for validation. The validation and dependability assessment of the fault tolerant strategy has been done using fault injection at the input output interfaces or within the inverted pendulum controller.

The dependability assessment exercise shows that design and programming mistakes like false gains or wrong polarity (false signs) leading to mission failures can be tolerated by proposed aspect oriented fault tolerant framework. Moreover input or output interface faults are also tolerated by avoiding partial (steady state errors) or complete failures using the proposed strategy. It has also been observed that faults introduced at the output interface have large error detection latency as compared to faults introduced at the input interface.

Although the proposed approach is focused on a mathematical model of inverted pendulum system, yet the principles derived and a general aspect oriented fault tolerant framework proposed should be applicable to other real time mission/safety critical controllers without much difficulty. For example the proposed aspect oriented fault tolerant framework dictates architecture to modularize error detection, error recovery or masking strategies in a non intrusive way. This can be applied irrespective of any complex error detection mechanism. Moreover the switch over logic can be scaled up to more variants as well.

The proposed error detection mechanism is dependent on a linearized model of system to be served. It is not possible to attain a very clean 100% true model of a system. However the safety check envelope may be relaxed to cover modeling uncertainties. Moreover the safety region based error detection (Lyupanov Stability Criterion) approach used in this work is only suitable for Linear Time Invariant dynamical mission or safety critical systems.

7. References

[1] Lara L. Pullum,"Software Fault Tolerance Techniques and Implementation", Artech House Inc., 2001.

[2] HORNING, J. J., et al., "A Program Structure for Error Detection and Recovery," in E. Gelenbe and C. Kaiser (eds.), Lecture Notes in Computer Science, Vol. 16, New York: Springer-Verlag, 1974, pp. 171–187.

[3] RANDELL, B., "System Structure for Software Fault Tolerance," IEEE Transactions on Software Engineering, Vol. SE-1, No. 2, 1975, pp. 220–232.

[4] HECHT, M., AND H. HECHT, "Fault Tolerant Software Modules for SIFT," SoHaR, Inc. Report TR-81-04, April 1981.

[5] ELMENDORF, W. R., "Fault-Tolerant Programming," Proceedings of FTCS-2, Newton, MA, 1972, pp. 79–83.

[6] AVIZIENIS, A., "On the Implementation of N-Version Programming for Software Fault-Tolerance during Execution," COMPSAC '77,

[7] CHEN, L., and A. AVIZIENIS, "N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation," Proceedings of FTCS-8, Toulouse, France, 1978, pp. 3–9.

[8] Kim, K. H. and H. O. Welch, "Distributed Execution of Recovery Blocks: An Approach for Uniform Treatment of Hardware and Software Faults in Real-Time Applications", IEEE Transactions on Computers, Vol. 38, No. 5, 1989, pp. 626-636.

[9] Scott, et al.," The Consensus Recovery Block", Proceedings of the Total Systems Reliability Symposium, 1985, pp. 74-85.

[10] L. Sha., "Dependable system upgrade", In *Proceedings of the IEEE Real-Time Systems Symposium*, page 440, IEEE Computer Society, 1998.

[11] M. Gopal, "Control Systems: Principles and Design", McGraw-Hill Inc., 2008.

[12] Marvin Bugeja,"Non-Linear Swing-Up and Stabilizing Control of an Inverted Pendulum System", In EUROCON 2003: Computer as a Tool, 22-24 Sept. 2003.

[13] Ghazi S.S.M , Jalali A.A.," Low Frequencies Optimal Control of an Inverted Pendulum", International Conference on E-Learning in Industrial Electronics, 18-20 Dec. 2006.

[14] K. Ogata, "*Modern Control Engineering"*, Prentice Hall, USA, 1998.

[15] D. Seto and L. Sha, "Engineering method for safety region development," Carnegie Mellon University/Software Engineering Institute, Tech. Report CMU/SEI-99-TR-018, Aug. 1999.

[16] Stephen Boyd, Michael Grant, "CVX: Matlab software for disciplined convex programming", version 1.2 (build 710), available on-line: http://stanford.edu/~boyd/cvx, accessed: December 7, 2008,

[17] S. Boyd, L. E. Ghaoul, E. Feron, and V. Balakrishnan, "*Linear Matrix Inequality in Systems and Control Theory*", Philadelphia, PA: SIAM Studies in Applied Mathematics, 1997, pp. 70

[18] J. Xu, B. Randell and A. Romanovsky. "A Generic Approach to Structuring and Implementing Complex Fault-Tolerant Software", Proceedings of the Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC.02), 2002.

[19] AspectC++ project homepage: http://www.aspectc.org

[20] Domen V. and Rok O., "Extension to Matlab/Simulink for design and implementation of distributed fault-tolerant control systems", In ICEEE, 2004.

[21] Mathworks, "Control flow of an S-Function", Available at:http://www.mathworks.com/company/newsletters/digest/sept98/controlflow.html, dated: 11-6-2008.

[22] Eliane Martins, Amanda C.A.Rosa, "A Fault Injection Approach Based on Reflective Programming", In Proceedings of Dependable Systems and Networks (DSN), 2000.

[23] Henrique Madeira, Diamantino Costa Marco Vieira, "On the Emulation of Software Faults by Software Fault Injection", In Proceedings of Dependable Systems and Networks (DSN), 2000.

[24] D. Theilliol et al.," Actuator Fault Tolerant Control Design Based on a Reconfigurable Reference Input", Int. J. Applied Mathematics Computer Science, 2008, Vol. 18, No. 4, 553–560.

[25] Kerrigan, E.C. and Maciejowski, J.M., "Fault-Tolerant Control of a Ship Propulsion System Using Model Predictive Control", In Proceedings of European Control Conference, 1999.

[26] Theilliol D., Noura H. and Ponsart J.C.,"Fault diagnosis and accommodation of a three-tank system based on analytical redundancy", *ISA Transactions 41(3), 2002, pp. 365–382.*