2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

A Reconfigurable Direct Control Allocation Method

Asim Kr Naskar, Sourav Patra and Siddhartha Sen

Abstract— **In the event of failure of actuators in an overactuated system a reconfigurable law is required to redistribute and coordinate the control effort among the remaining healthy actuators. In this paper, a modified lookup table based Direct Control Allocation method is proposed for reconfiguration of actuator command, for a single actuator fault, without updating the lookup tables. The proposed method saves both memory as well as time to prepare the tables. The method of reconfigurable Direct Control Allocation is then applied to a satellite launch vehicle model. The simulation results show satisfactory performance of the proposed method in the presence of a single Stuck-at-One type actuator fault.**

I. Introduction

The survivability of air vehicles depends on the ability of the system to tolerate faults, such as actuator fault, sensor fault etc. Actuator fault may be of different types, but the most vulnerable one is Stuck-at-One actuator fault. In case of such fault, faulty actuators no longer respond to the control input but generate a constant output which can be regarded as a constant disturbance input to the system. So, faulty actuators not only reduce the flexibility to control the system but also act as a source of persistent external disturbance. The Fault Tolerant Control (FTC) is to compensate for the effect generated by the faulty actuators maintaining the original performance of the system as much as possible using remaining (healthy) actuators. For the FTC, the system requires redundant actuators and a reconfigurable control law which compensates for the effect of failure or damage of actuators. A system with redundant actuators, i.e., an overactuated system can achieve FTC by reconfiguration of controller [1], [2], [3], [4], [5], or by reconfiguration of allocation of actuators [4], [6], [7], [8], [9]. Different methods for control allocation found in the literature can broadly be categorized in two types: direct methods [10], [11] and optimization based methods [12], [13], [14], [15]. Comparing to optimization based methods, direct methods (or direct control allocation methods) are computationally efficient for specific applications.

The Direct Control Allocation (DCA) method finds a one-to-one relation between an actuator command and the controller command (for flight control applications, it is termed as a moment vector) [10]. In the DCA method, actuator position constraint is taken into account, and using those constraints moment set is pre-calculated, denoted by Attainable Moment Set (AMS). The boundary of the AMS is used to generate actuator command. By introduction of spherical coordinate, the information from three dimensional AMS boundary is converted to a set of three tables. Through this conversion, the computational burden of the method is reduced considerably, which makes it suitable for online application [11]. The modified method is called lookup table based direct control allocation. However, the technique has been used for healthy actuators only.

The reconfigurability of allocation or reallocation can be considered as an essential property of a good allocation method to maintain stability and reliability of overactuated systems. Burken et al. [2] have studied a reconfigurable flight control technique using Fixed Point Quadratic Programming method. Some other approaches are available in [6], [7], [8] and [9].

In the present work, we have addressed the problem of reconfiguration of actuators using lookup table based DCA method. A lookup table based method is chosen for this work, since it has a negligible online computational burden in healthy condition. In the event of a Stuck-at-One fault, the role of the proposed control allocation algorithm is to redistribute the command so as to nullify the effect of the fault and simultaneously generate the desired moment by the healthy actuators without updating the original lookup tables. This is done in an iterative manner as elaborated in Section IV. It is assumed that a fault detection and isolation unit is present in the system.

The rest of the paper is organized as follows. Section II formulates the problem, while Section III describes the background for the lookup table based direct control allocation method. Section IV presents the method of reconfiguration, the main contribution of this paper, and in Section V simulation results are presented. Finally Section VI concludes the paper.

II. Problem Description

To demonstrate the actuator command reallocation in a fault-tolerant overactuated flight control system, a basic feedback structure of the system is shown in Fig. 1. For fault tolerant control, the system needs a stabilizing controller and a reconfigurable allocation module. It is assumed that a Fault Detection and Isolation (FDI) unit is present in the system. In Fig. 1, $\overline{r}$ is the reference command from the pilot or autopilot, $\overline{v}_d \in \mathbb{R}^3$ is the desired moment, $\overline{v} \in \mathbb{R}^3$ is the actual moment generated by the actuators and $\overline{\delta} \in \mathbb{R}^p$, where $p > 3$ for overactuated systems, is the actuator command from the allocator.

Let the time-invariant linearized state dynamics of an air vehicle be given by

$$\dot{\overline{x}} = A\overline{x} + G\overline{v}, \tag{1}$$

Authors are with the Electrical Engineering Dept., Indian Institute of Technology Kharagpur, 721302, India. Tel.: +91-3222-283084; fax: +91-3222-22262. E-mail: asim, sourav, ssen@ee.iitkgp.ernet.in

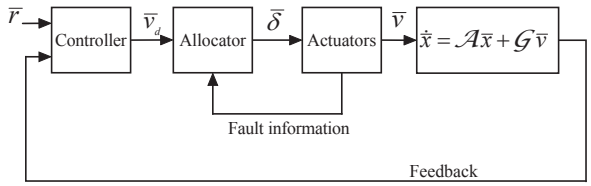

Fig. 1. Basic control block of an air vehicle with allocator

$$\overline{v} = B\overline{\delta}, \tag{2}$$

where, $\overline{\delta}_{i,\min} \leq \overline{\delta}_i \leq \overline{\delta}_{i,\max}$, for $i = 1, 2, ..., p$, and $\overline{x} \in \mathbb{R}^n$ is the state vector. $A \in \mathbb{R}^{n\times n}$, $G \in \mathbb{R}^{n\times 3}$ and $B \in \mathbb{R}^{3\times p}$ are the system matrix, the equivalent input matrix and the control effectiveness matrix, respectively. $\overline{v} = \overline{v}_d$ while the actuators generate $\overline{v}_d$ perfectly.

Suppose, a Stuck-at-One fault has occurred in l-th actuator, and the actuator is stuck at $\delta_{l,\max}$ and produces a constant disturbance moment $\overline{d}_l = \overline{b}_l\delta_{l_{\max}}$, where $\overline{b}_l$ is the l-th column of B. So, (2) can be rewritten as

$$\overline{v} - \overline{d}_l = B\overline{\delta}^l \tag{3}$$

where $\overline{\delta}^l \in \mathbb{R}^p$ with l-th element is equal to zero.

Now the problem can be stated as follow: In the event of a Stuck-at-One fault in l-th actuator, using DCA produce the vector $\overline{\delta}^l$ that minimize the difference (in the sense of 2-norm) between the left hand side and the right hand side of (3) for given B and $\overline{v} = \overline{v}_d$ maintaining the actuator constraints $\overline{\delta}_{i,min} \leq \overline{\delta}^l_i \leq \overline{\delta}_{i,max}$ where $i = 1, 2, ..., p$.

III. Direct Control Allocation Using AMS

The direct control allocation method using AMS for three-dimensional moment problems has been described in details in [11]. For a p-actuator system the actuator command vector is confined to a p-dimensional hyper rectangular volume generated by the actuator position constraints, and an AMS is obtained by linear transformation of that volume through the control effectiveness matrix of the system, $B \in \mathbb{R}^{3\times p}$. In other words, AMS is the set of all possible three-dimensional moments that can be generated by the actuators obeying their position constraints.

A. AMS formation

Depending on the property of the control effectiveness matrix, a system can be categorized into two types [11]. If any three or more columns of the control effectiveness matrix, B, are linearly dependent (in a plane), the system is called a coplanar system; otherwise, it is called a non-coplanar system. If a system is of the latter type, its AMS has parallelogram plane facets on the boundary, and every point on the boundary corresponds to a unique actuator command. In the rest of the paper, we consider the system as non-coplanar, unless otherwise stated.

A pair of actuators decides the size of a pair of parallelogram facets placed opposite to each other on the AMS. These two facets are named as Max and Min facets [11]. In this paper, for i-th and j-th actuators we denote Max and Min facets by $facet_{1,ij}$ and $facet_{2,ij}$ respectively. Fig. 2 shows a facet, $facet_{1,ij}$, where two sides connected to a vertex are denoted by vectors $\overline{\Delta m}_i \in \mathbb{R}^3$ and $\overline{\Delta m}_j \in \mathbb{R}^3$, and the vector from the origin, O, to that vertex is denoted by $\overline{M}_{base1,ij} \in \mathbb{R}^3$. Any moment vector on the facet can be represented by these three *independent vectors*. Let we consider a moment vector $\overline{v}_d$ and a facet $facet_{1,ij}$. Then one may write $\overline{v}_d = \alpha_1\overline{M}_{base,1ij} + \alpha_2\overline{\Delta m}_i + \alpha_3\overline{\Delta m}_j$, or

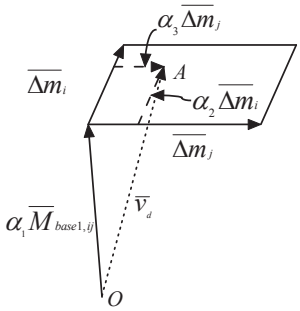

Fig. 2. Facet, $facet_{1,ij}$, in three dimensional moment space.

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} \overline{M}_{base1,ij} & \overline{\Delta m}_i & \overline{\Delta m}_j \end{bmatrix}^{-1} \overline{v}_d \tag{4}$$

Next, how the values obtained from (4) can be used to determine whether a vector $\overline{v}_d$ is cutting a facet or pointing or not pointing to a facet, $facet_{1,ij}$, is explained. Consider the following conditions

C1 $\alpha_1 > 0$,
C2 $0 < \alpha_2/\alpha_1,\ \alpha_3/\alpha_1 \leq 1$.

Now the following conclusions can be made.

1) If (C1) and (C2) are satisfied,
   - R1 Extended $\overline{v}_d$ intersects the facet, when $\alpha_1 < 1$.
   - R2 $\overline{v}_d$ touches the facet, when $\alpha_1 = 1$.
   - R3 $\overline{v}_d$ intersects the facet, when $\alpha_1 > 1$.
2) If only (C1) is satisfied but not (C2),
   - R4 Extended $\overline{v}_d$ intersects the *extended facet*, when $\alpha_1 < 1$.
   - R5 $\overline{v}_d$ touches the *extended facet*, when $\alpha_1 = 1$.
   - R6 $\overline{v}_d$ intersects the *extended facet*, when $\alpha_1 > 1$.
3) If (C1) is not satisfied,
   - R7 $\overline{v}_d$ directs opposite to the facet with respect to the origin.

The cases (R1) — (R6) will be used during the search for a facet in the proposed algorithm. Fig. 2 shows the case (R2) when $\overline{v}_d$ is touching the facet $facet_{1,ij}$.

For $facet_{1,ij}$, the vectors $\overline{M}_{base1,ij}$, $\overline{\Delta m}_i$ and $\overline{\Delta m}_j$ are defined as follows.

$$\left.\begin{aligned} \overline{M}_{base1,ij} &= \overline{M}_{d1,ij} + \overline{b}_i\delta_{i,\min} + \overline{b}_j\delta_{j,\min}, \\ \overline{\Delta m}_i &= (\delta_{i,\max} - \delta_{i,\min})\overline{b}_i, \\ \overline{\Delta m}_j &= (\delta_{j,\max} - \delta_{j,\min})\overline{b}_j, \end{aligned}\right\} \tag{5}$$

where $\overline{M}_{d1,ij}$ is a positioning vector (defined later), $\overline{b}_i$, $\overline{b}_j$ are respectively i-th and j-th column of B. For $facet_{2,ij}$ the

side vectors, $\overline{\Delta m_i}$ and $\overline{\Delta m_j}$, are same but the base vector is $\overline{M}_{base2,ij} = \overline{M}_{d2,ij} + \overline{b}_i\delta_{i,\min} + \overline{b}_j\delta_{j,\min}$.

Now, $\overline{M}_{d1,ij}$ and $\overline{M}_{d2,ij}$ are defined as [11]:

$$\left.\begin{aligned} &\overline{M}_{d1,ij} = \sum_{k=1,\, k\neq i,j}^{p} \overline{\mu}_{k,\max}, \\ &\text{where } \overline{\mu}_{k,\max} = \begin{cases} \overline{b}_k\delta_{k,\max} & \text{if} \quad \overline{b}_k^T\overline{\eta}_{ij} > 0 \\ \overline{b}_k\delta_{k,\min} & \text{if} \quad \overline{b}_k^T\overline{\eta}_{ij} < 0. \end{cases} \end{aligned}\right\} \quad (6)$$

$$\left.\begin{aligned} &\overline{M}_{d2,ij} = \sum_{k=1,\, k\neq i,j}^{p} \overline{\mu}_{k,\min}, \\ &\text{where } \overline{\mu}_{k,\min} = \begin{cases} \overline{b}_k\delta_{k,\max} & \text{if} \quad \overline{b}_k^T\overline{\eta}_{ij} < 0 \\ \overline{b}_k\delta_{k,\min} & \text{if} \quad \overline{b}_k^T\overline{\eta}_{ij} > 0. \end{cases} \end{aligned}\right\} \quad (7)$$

$\overline{\eta}_{ij} = \overline{b}_i \times \overline{b}_j$ is a normal vector to $facet_{1,ij}$. From (6) and (7), it is apparent that Max and Min facets are placed opposite to each other on the AMS.

B. Offline Array formation

All the information from (5), (6) and (7) is stored in arrays as follows: Array-1 keeps the facet number of the points (on the AMS) whose position is determined by the azimuth and horizontal angle of that point. Array-2, a one dimensional array, keeps the set of vectors in (5) for each facet. Suppose, the set of commands δ_k, where $k = 1, 2, \ldots, p$, used in (6) is stored in a p-dimensional vector $\overline{\Delta}_{1,ij}$. The third array, Array-3, is also a one-dimensional array that stores vectors like $\overline{\Delta}_{1,ij}$ for each facet. Note that i-th and j-th component of the stored vector $\overline{\Delta}_{1,ij}$ are undecided where all others have fixed values (limiting values). The above two components are called free commands that are to be calculated using (4) and following the procedure described in Sec. III.A.3 of [11]. For detail information on the offline-array formation, interested readers are referred to [11].

IV. Reconfiguration

In the event of an actuator failure the pre-fault AMS will be changed, and to generate an actuator command after failure the post-fault AMS information is required. But the reconstruction of new arrays is costly in computation. Now (5), (6) and (7) reveal that the changes in the pre-fault AMS after the fault is as follows. Some facets will be lost (called faulty facets) and the rest of the facets (called non-faulty facets) will translate by a fixed vector length to form the post-fault AMS inside the pre-fault AMS. Hence corresponding to a non-faulty facet $facet_{1,ij}$ on the outer AMS (or pre-fault AMS) a similar facet, denoted by $facet'_{1,ij}$, will be obtained on the inner AMS (or post-fault AMS). The vector by which two similar facets are translated is called fault vector. For the l-th actuator fault, the fault vector is denoted by $\overline{F}_l$.

For $facet'_{1,ij}$ the set of vectors similar to (5) can be defined as

$$\left.\begin{aligned} \overline{M}'_{base1,ij} &= \overline{M}_{base1,ij} - \overline{F}_l, \\ \overline{\Delta m}'_i &= \overline{\Delta m}_i, \\ \overline{\Delta m}'_j &= \overline{\Delta m}_j, \end{aligned}\right\} \quad (8)$$

and a vector $\overline{v}_d$ can be represented by the set of vectors as

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} \left(\overline{M}_{base1,ij} - \overline{F}_l\right) & \overline{\Delta m}_i & \overline{\Delta m}_j \end{bmatrix}^{-1} \overline{v}_d. \quad (9)$$

In this paper, a new method is developed to reallocate the actuator command in the event of a single actuator fault using (8), (9) and the information available in pre-fault arrays. In what follows, the method is explained with the help of a two-dimensional moment problem.

A. Two-dimensional moment space

In this section, we illustrate the method of reconfiguration with a two dimensional example. For simplicity, let the fault be a Stuck-at-Zero fault (i.e., the faulty actuator generates zero moment). In this case, the only task is to redistribute a control effort generated by the controller among the healthy actuators. Let the control effectiveness matrix $B = \begin{bmatrix} 10 & 18 & 0 & 8 & 3 & 5 \\ 2 & -2 & -3 & 20 & -3 & 6 \end{bmatrix}$, and the actuator constraints be given by $\overline{\delta}_{\max} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$ and $\overline{\delta}_{\min} = -\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$. There are six actuators

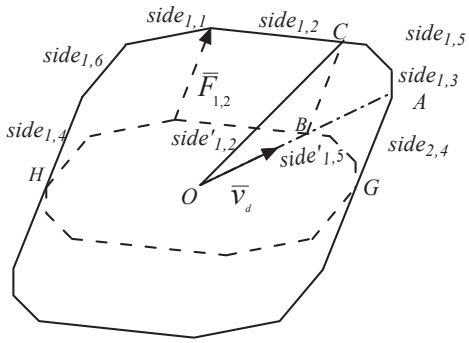

Fig. 3. The solid-line polygon is pre-fault AMS and the dotted-line polygon is post-fault AMS.

that constitute an AMS in the form of a convex polygon in two-dimensional space. In Fig. 3, the outer solid-line boundary with 12 sides (analogous to facets in a three dimensional AMS) is the pre-fault AMS with six healthy actuators. Whereas, the inner dotted-line boundary having 10 sides is the post-fault AMS with five actuators when a Stuck-at-Zero fault occurs in the 4-th actuator (i.e., $l = 4$). Due to a fault in a single actuator, the pre-fault AMS polygon loses two sides. Here, $side_{2,4}$ and $side_{1,4}$ (faulty) collapse to the points G and H, respectively, and other sides (non-faulty) translate all together by a length $\overline{F}_4$ that in turn form the inner AMS boundary. In Fig. 3, $side'_{1,2}$ is similar to $side_{1,2}$ and they are separated by the fault vector $\overline{F}_4$.

In general, if the fault vector $\overline{F}_l$ is known, analogous to (9), a vector $\overline{v}_d$ in two-dimensional cases can be represented as

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \left(\overline{M}_{base1,i} - \overline{F}_l\right) & \overline{\Delta m}_i \end{bmatrix}^{-1} \overline{v}_d. \quad (10)$$

Suppose, a desired moment vector $\overline{v}_d$ points at A on $side_{1,3}$ and at B on $side'_{1,2}$ as shown in Fig. 3. Since $side'_{1,2}$,

similar to $side_{1,2}$, translates along $-\overline{F}_4$ there exists a point C on $side_{1,2}$ such that $\overline{BC} = \overline{F}_4$. The reconfiguration task is to find $side_{1,2}$ or the point C (instead of the point B) on the outer AMS boundary from the knowledge of $\overline{F}_4$, the point A and the pre-fault arrays. The method will be described in the next subsection.

B. The proposed algorithm

The proposed algorithm is explained for two-dimensional systems. Suppose, $\overline{v}_d$ points at A on the outer AMS boundary and at B on the inner AMS boundary. The point C on the outer AMS is such a point that $\overline{F}_l = \overline{BC}$. $\overline{v}_d$ has to be generated by a set of $p-1$ actuators with a Stuck-at-Zero fault in l-th actuator. For searching B, a moment vector is defined as $\overline{v}_m = \beta\overline{OA} + \overline{F}_l$, where β is a scalar. The proposed algorithm starts with $\beta = 1$, and during the iteration β is updated. It can be proved that β monotonically decreases up to the value $\frac{|\overline{OB}|}{|\overline{OA}|}$ where the iteration stops.

Inputs to algorithm

- lookup tables for pre-fault AMS: Array-1, Array-2 and Array-3.
- Information of faulty actuator (single) from FDI unit.
- Desired vector $\overline{v}_d$.

Solution algorithm

Step 1. Calculate the fault vector $\overline{F_l}$: Find the *side* from Array-1 where $\overline{v}_d$ points at. Pick the vector $\overline{\Delta}$ from Array-3 for the above *side*. Store the l-th component of $\overline{\Delta}$ in a scalar variable γ.
(a) If γ contains a fixed control: (see the Arrays formation in Sec. III.B) $\overline{F}_l = \overline{b}_l\gamma$.
(b) If γ contains a free control: Recalculate γ as follows. First, calculate α_1 and α_2 in (10) using the *set of vectors* from Array-2 and $\overline{F_l} = 0$. With those values, calculate δ_l using the procedure described in Sec. III.A.3 of [11]. Now $\gamma = \delta_{l,max}$ if $\delta_l = +ve$, otherwise, $\gamma = \delta_{l,min}$. $\overline{F}_l = \overline{b}_l\gamma$.

Step 2. Initialization: $\beta = 1$, $k = 1$. Use the *side* information from Step-1 to pick the *set of vectors* from Array-2 and calculate $\overline{OA}$ using (10).

Step 3. Calculate $\overline{v}_m = \beta\overline{OA} + \overline{F}_l$. Find the *side* from Array-1 where $\overline{v}_m$ points at. Pick the *set of vectors* from Array-2 using the *side* information, and calculate α_1 and α_2 in (10) (i.e., finding intersection of $\overline{v}_m$ by a similar side).

Step 4. If $0 < \alpha_2/\alpha_1 \leq 1$ is satisfied (see (R1) — (R6)) do the following: Pick $\overline{\Delta}$ from Array-3 using the *side* calculated in Step-3. Calculate and store the free controls in $\overline{\Delta}$ following the procedure described in Sec. III.A.3 of [11] using α_1, α_2 and the *set of vectors* from Step-3. $\overline{\Delta}$ is stored in $\overline{\delta}$. **Go to Step-6**.

Step 5. Calculate $\overline{OB_k} = \overline{OA}/\alpha_1$ where α_1 is obtained from Step-3, and $\beta = \frac{|\overline{OB}_k|}{|\overline{OA}|}$. Update $k = k + 1$ and **go to Step-3**.

Step 6. Set $\delta_l = 0$ in $\overline{\delta}$ and denote it by $\overline{\delta}^l$ (Note that, the command vectors $\overline{\delta}$ and $\overline{\delta}^l$ correspond to the moment vectors $\overline{OC}$ and $\overline{OB}$ respectively).

Output from algorithm

The reconfigured actuator command is $\overline{\delta}^l$.

Due to the space constraint, the proof of monotonicity or in other words the convergence of the algorithm is not given here and it will be published elsewhere. Instead, the procedure for finding the point B following the steps given in the proposed algorithm is illustrated using two case studies.

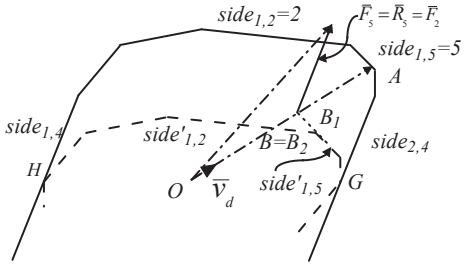

Fig. 4. A part of the AMS shown in Fig. 3 with $\overline{v}_d$ pointing to a non-faulty side.

Case 1: In Fig. 4, consider the desired vector $\overline{v}_d$ points at A on a non-faulty side $side_{1,3}$. The fault has occurred in actuator-4. The fault vector $\overline{F}_4$ is calculated in Step-1 (a). The iteration starts with $k = 1$ and $\beta = 1$ (in Step - 2). In Step-3, the vector $\overline{v}_m = \overline{OD}$ intersects $side_{1,5}$. The extension of $side'_{1,5}$ intersects $\overline{OA}$ at the point B_1 which implies that the values of α_1, α_2 (from Step-3) are such that the condition in Step-4 fails (see (R4) — (R6)). Then β is calculated in Step-5. In the next iteration, $k = 2$, the vector $\overline{v}_m = \beta\overline{OA} + \overline{F}_4 = \overline{OB_1} + \overline{F}_4 = \overline{OD'}$ intersects $side_{1,2}$, whereas $side'_{1,2}$ intersects $\overline{OA}$ at B. The condition in Step-3 is satisfied. So, the the iteration ends in two steps. The values of β for $k = 1$ and 2 are respectively $\frac{|\overline{OB}_1|}{|\overline{OA}|}$ and $\frac{|\overline{OB}|}{|\overline{OA}|}$. So β decreases in consecutive steps.

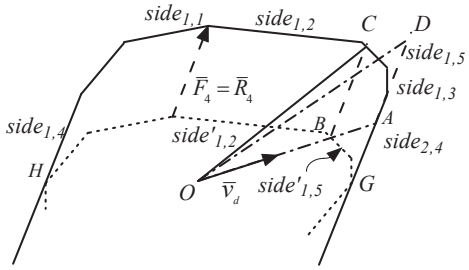

Fig. 5. A part of the AMS shown in Fig. 3 with $\overline{v}_d$ pointing to a faulty side.

Case 2: Consider Fig. 5 where the desired moment vector $\overline{v}_d$ points to a faulty side, $side_{2,4}$. The vector $\overline{F}_4$ is calculated in Step-1 (b) of the algorithm as $side_{2,4}$ is a faulty side. During the first iteration, $k = 1$, the vector $\overline{v}_m = \overline{OD}$ intersects $side_{1,5}$; and $side'_{1,5}$ intersects $\overline{OA}$ at the point B and the iteration ends. In this case, one iteration is required to search the point B.

C. Three-dimensional moment space—Non-coplanar systems

The modification required to adapt the above algorithm for a three-dimensional moment space are as follows: Instead of using (10) and the ratio α_2/α_1 use (9) and the ratios α_2/α_1 and α_3/α_1 both to decide the position of a point on a facet.

D. Three-dimensional moment space—Coplanar Systems

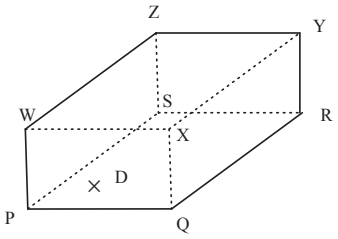

Fig. 6. A two-dimensional hexagonal facet PQRYZW and its sub-facets (shown by dotted lines) for a group of three coplanar actuators.

For coplanar systems, some AMS boundary facets may be polygonal and they consist of many overlapping parallelogram facets called sub-facets [11]. In Fig. 6, PQRYZW is a hexagonal facet generated by a set of three coplanar actuators, and the dotted lines inside the hexagon shows the sub-facets. For coplanar systems, Array-1 is modified, as described in [11], to keep records of all the sub-facet numbers along the third dimension of the array. To accommodate this change, Step-3 of the algorithm is modified to include a search in the third direction of Array-1 to find for a non-faulty facet (because only a non-faulty facet has a similar facet on the inner AMS boundary) whenever a vector points at a polygonal facet. In Fig. 6, suppose a vector points at D (marked by a dot) on the hexagonal facet. The point D is on the faulty sub-facet $PQRS$ as well as on the non-faulty sub-facet $PQXW$, and by the additional search in Step-3 of the algorithm it is required to know the facet-number of the non-faulty sub-facet $PQXW$.

E. Reconfiguration for Stuck-at-One type fault

As we have assumed that an FDI unit is present in the system, the disturbance moment generated by the faulty actuator can be calculated as $\overline{d}_l = \overline{b}_l \delta_l$. The desired moment vector is modified by subtracting the calculated disturbance as $\overline{v}_T = \overline{v}_d - \overline{d}_l$. Now the task is to allocate $\overline{v}_T$ by healthy actuators considering the fault as Stuck-at-Zero fault and following the same procedure explained in Section IV. A . If the rest of the healthy actuators can produce the modified moment vector $\overline{v}_T$, then the closed loop performance will be unaltered.

V. Simulation Results

The proposed method of reconfiguration is applied to control allocation of a satellite launch vehicle model [16]. The control effectiveness matrix of the system is B (given in Appendix). The actuator constraints are $\overline{\delta}_{\max} = \begin{bmatrix} 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 \end{bmatrix}^T$ deg and $\overline{\delta}_{\min} = -\begin{bmatrix} 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 \end{bmatrix}^T$ deg. The launch vehicle is a coplanar system. Moreover, the actuators 5 and 7 are co-linear, and the facets contributed by them are merely a line, and so they are not considered as sub-facets. Here the number of facets are 122 (including sub-facets).

The open loop dynamics of the launch vehicle model are unstable, and it is made stable by a full state feedback [16]. The static state feedback matrix F and system matrices A and G are given in Appendix.

The simulation started with the pitch, roll and yaw angle command of 0.5 deg, 0.8 deg and 0.3 deg respectively. After that a step command for pitch angle of 2.5 deg was given at 1 sec. The fault developed in actuator-2 when the actuator

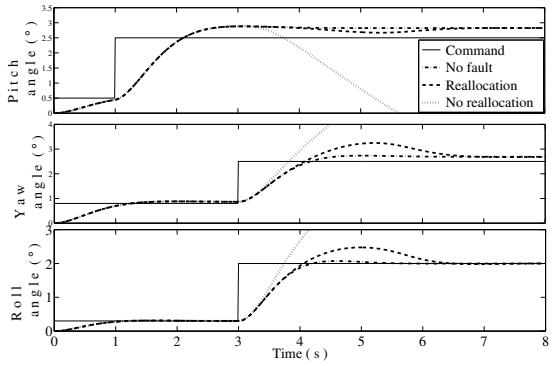

Fig. 7. Pitch, Yaw, and Roll positions in three different conditions: No fault, reallocation after a Stuck-at-One fault in actuator-2 and no reallocation after the fault— for the step reference commands .

had reached its positive saturation after the application of step commands of 2 deg and 2.5 deg for roll and yaw angle, respectively, at 3 sec. All the responses before and after the fault are shown in the Figs. 7-9.

In Fig. 7, it is shown that without reallocation the system generates unbounded output. But the system response after

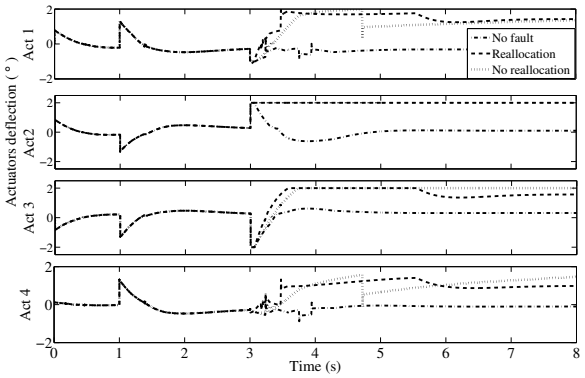

Fig. 8. Deflection of actuators 1-4 in three different conditions: No fault, reallocation after a Stuck-at-One fault in actuator-2 and no reallocation after the fault.

the fault with reallocation (by the proposed method) follows the no-fault response in steady state. The initial overshoot after the fault is due to the saturation in more than one actuator. In Figs. 8-9, actuators' responses are shown. The position of actuator-2 after 3 sec is constant at 2 deg as

it has a Stuck-at-One fault. All other actuators, without

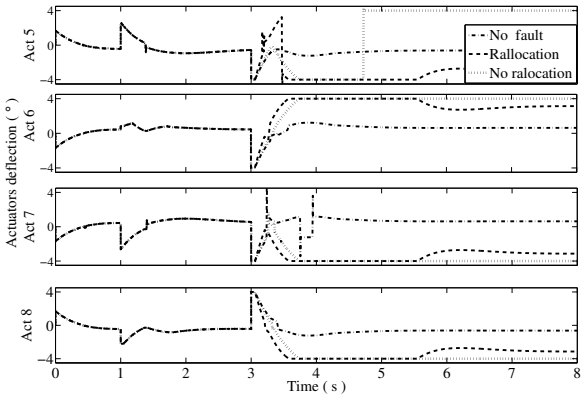

Fig. 9. Deflection of actuators 5-8 in three different conditions: No fault, reallocation after a Stuck-at-One fault in actuator-2 and no reallocation after the fault.

reallocation, are gradually driven to the position saturation limits. But, if the actuator command is reallocated, actuators are positioned away from the saturation limits according to the moment (generated by the faulty actuator) demand.

VI. Conclusions

The AMS based direct control allocation is a powerful method for distributing control commands in an overactuated system. The attainable moment set is prepared a priori and stored in form of lookup tables from which allocation can be made directly. But in the event of fault in an actuator the moment set gets disturbed and usually fresh lookup tables need to be prepared. *In the present paper, a new iterative scheme is proposed where the reallocation in the post-fault scenario is performed using the original lookup tables itself. The algorithm is suitable to tackle both Stuck-at-Zero and Stuck-at-One faults*. The scheme considers only one actuator failure and assumes that the fault information is known a priori through a fault detection scheme. The efficacy of the method is explained with reference to the control allocation problem of an eight-actuator satellite launch vehicle model. The computational overhead is minimal and it has been observed that the algorithm normally requires two/three iterations before convergence and hence can effectively be used for online applications.

References

[1] W. D. Morse and K. A. Ossman, Model following reconfigurable flight control system for the AFTI/F-16, Journal of Guidance, Control, and Dynamics, vol. 13, no. 6, pp. 969-976, 1990.

[2] J. J. Burken, P. Lu, Z. Wu, and C. Bahm, Two reconfigurable flight-control design methods: robust servomechanism and control allocation, Journal of Guidance, Control, and Dynamics, vol. 24, no. 3, pp. 482-493, 2001.

[3] W. Chen and J. Jiang, Fault-tolerant control against stuck actuator faults, IEE Proceedings Control Theory and Applications, vol. 152, no. 2, 2005, pp. 138-146.

[4] R. K. Das, S. Sen, and S. Dasgupta, Robust and fault tolerant controller for attitude control of a satellite launch vehicle, IET Control Theory and Applications, vol. 1, no. 1, pp. 304-312, 2007.

[5] S. Ye, Y. Zhang, X. Wang and C. A. Rabbath, Robust fault-tolerant control using on-line control re-allocation with application to aircraft, American Control Conference, St. Louis, USA, 2009, pp. 5534-5539.

[6] M. N. Demenkov, Reconfigurable linear control allocation via generalized bisection, in AIAA 2005-6341, 2005.

[7] M. Demenkov, Reconfigurable direct control allocation for overactuated systems, IFAC Proceedings Volumes (IFAC-Papers Online), vol. 18 (part 1), 2011, pp. 4696-4700.

[8] H. Wang, J. Yi, and G. Fan, Autonomous reconfigurable flight control system design using control allocation, in 2nd International Symposium on Systems and Control in Aerospace and Astronautics, 2008, pp. 1-6.

[9] Y. Zhong, L. Yang, and G. Shen, Control allocation bsaed reconfigurable flight control for aircraft with multiple control effectors, in AIAA 2009-58, 2009.

[10] W. C. Durham, Constrained control allocation, Journal of Guidance, Control, and Dynamics, vol. 16, no. 4, pp. 717-725, 1993.

[11] J. Petersen and M. Bodson, Fast implementation of direct allocation with extension to coplanar controls, Journal of Guidance, Control, and Dynamics, vol. 25, no. 3, pp. 464-473, 2002.

[12] M. Bodson, Evaluation of optimization methods for control allocation, Journal of Guidance, Control, and Dynamics, vol. 25, no. 4, pp.703-711, 2002.

[13] O. Harkegard, Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation, in Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, USA, Dec. 2002, pp. 1295-1300.

[14] M. Bodson and S. A. Frost, Load balancing in control allocation, Journal of Guidance, Control, and Dynamics, vol. 34, no. 2, pp. 380-387, 2011.

[15] L. Cui and Y. Yang, Disturbance rejection and robust least-squares control allocation in flight control system, Journal of Guidance, Control, and Dynamics, vol. 34, no. 6, pp. 1632-1643, 2011.

[16] W. C. A. Kishore, S. Sen and G. Ray, Disturbance rejection and control allocation of over-actuated systems, in Proceedings of the IEEE International Conference on Industrial Technology, 2006, pp. 1054-1059.

Appendix

$$B = \begin{bmatrix} 0.2985 & -0.2985 & -0.2985 & 0.2985 \\ -0.2985 & -0.2985 & 0.2985 & 0.2985 \\ -0.1618 & 0.1618 & -0.1618 & 0.1618 \\ 0.0779 & 0 & -0.0779 & 0 \\ 0 & 0.0779 & 0 & -0.0779 \\ 0 & 0.0211 & 0 & 0.0211 \end{bmatrix} \frac{\pi}{180},$$

$$A = \begin{bmatrix} 0 & 1000 & 0 & 0 & 0 & 0 \\ 706.6 & 0 & 0.0187 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0.0271 & 0 & 437.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \\ -0.5713 & 0 & 0.5468 & 0 & 0 & 0 \end{bmatrix} \times 10^{-3},$$

$$G = \begin{bmatrix} 0 & 0 & 0 \\ 5750.8 & -0.2223 & -1.4948 \\ 0 & 0 & 0 \\ 0.2207 & -5173 & 2.0768 \\ 0 & 0 & 0 \\ -4.6494 & 8.784 & 14153 \end{bmatrix} \times 10^{-3},$$

$$F = \begin{bmatrix} 1069.3 & 620.84 & -0.0407 \\ 0.0378 & 0.0257 & -1207.7 \\ 0.2591 & 0.1957 & 0.8037 \\ -0.0272 & 0.0353 & 0.0560 \\ -709.47 & 0.2344 & 0.1163 \\ 0.4419 & 549.49 & 278.77 \end{bmatrix}.$$