A Modified Direct Allocation Algorithm with Application to Redundant Actuators

Article *in* Chinese Journal of Aeronautics · June 2011

DOI: 10.1016/S1000-9361(11)60035-6

3 authors, including:

Shijie Zhang
Harbin Institute of Technology
24 PUBLICATIONS **281** CITATIONS

SEE PROFILE

Yulin Zhang
Zhejiang University
133 PUBLICATIONS **1,315** CITATIONS

SEE PROFILE

Chinese Journal of Aeronautics 24 (2011) 299-308

Contents lists available at ScienceDirect

Chinese Journal of Aeronautics

journal homepage: www.elsevier.com/locate/cja

A Modified Direct Allocation Algorithm with Application to Redundant Actuators

TANG Shengyong[a], ZHANG Shijie[a,*], ZHANG Yulin[b]

[a]*Research Center of Satellite Technology, Harbin Institute of Technology, Harbin 150080, China*

[b]*College of Aerospace and Material Engineering, National University of Defense Technology, Changsha 410073, China*

Received 21 June 2010; revised 26 July 2010; accepted 17 November 2010

Abstract

Control allocation considers the problem of controlling instruction distribution for control systems with multiple and redundant actuators. This paper focuses on the direct allocation method, making the time requirement of the algorithm analogous compared with modified pseudoinverse redistribution methods, linear programming methods solved by simplex method, and sub-gradient optimization method. To reduce off-line computations of constructing the attainable moment set of actuators, a new approach based on the null space of the control effectiveness matrix is proposed, which is superior when the number of actuators is less than 10 compared with traditional method. To decrease on-line computations, an improvement method of searching the facet that is aligned with the desired moment is presented, shortening the search time by checking only the facets that lie around the desired moment. To find such facets, the vertices of the attainable moment set are normalized and saved during off-line computations. Simulation results show that at least 32.22% of off-line computation time would be saved using null space-based construction when the number of actuators is less than 10. In on-line computations, the modified method performs superiorly compared with the three aforementioned methods. Furthermore, it may solve the problem of control allocation efficiently when a remarkable large number of redundant actuators are configured.

Keywords: control allocation; direct allocation; pseudoinverse redistribution; linear programming; sub-gradient optimization; attainable moment set

1. Introduction

In order to increase the maneuver ability and reliability of spacecraft, configuration with redundant actuators is commonly considered. However, control requirements cannot be generated directly by the actuators because the configuration of redundant actuators causes the variety of commands that meet the requirements. Control allocation is the problem of distributing the control requirements among redundant actuators for satisfying the optimized objectives within their range of position and rate limits[1]. Great attention has been paid to the improvement of allocation algorithms[2-3], the application of the technique to engineering[4-5] and the extension to theoretical approaches[6] in the recent years.

This paper focuses on the improvement of allocation algorithms. The properties of evaluating an algorithm of control allocation would be: allocation error between the desired moment (DM) and the output moment generated by the actuators, controls that the algorithm consumes to generate a given DM, allocation subset that consists of the moments generated by the algorithm with the actuators within their position and

*Corresponding author. Tel.: +86-451-86402357-8409.
E-mail address: sjzhang@hit.edu.cn
Foundation items: National Natural Science Foundation of China (NSFC60704020); Changjiang Scholars and Innovative Research Team of China (PCSIRT0520); Research Fund for the Doctoral Program of Higher Education of China (20070213068)

doi: 10.1016/S1000-9361(11)60035-6

rate limits, memory space that the algorithm needs to save off-line data, and allocation time that the algorithm spends in calculating the controlling instruction of the actuators. The allocation time is split into off-line computation time and on-line computation time. Off-line computations calculate the unchanged data of the algorithm and save them in memory space. It may be performed when the configuration of actuators is designed or, in the case of a reconfigurable control law[7]. On-line computations calculate the commands of each actuator in real-time when given a DM. The allocation subset is evaluated in comparison with the attainable moment set (AMS) composed of all the moment vectors that can be generated by the actuators within their control constraints.

The allocation time, especially the on-line computation time, has a significant effect on the real-time application of an algorithm, while the other properties should also be considered for specific applications. The former allocation approaches (for example, daisy chaining, generalized inverse and pseudoinverse) use little memory space and their property of allocation time is attractive, but the existence of allocation error in AMS and the limited allocation subset (the volume is strictly less than that of AMS, and the ratio descends dramatically in high redundant problems[8]) restrict their applications. Modified pseudoinverse redistribution (MPIR)[2] methods and a sub-gradient optimization (SGO) method[5] have been proposed to enlarge the allocation subset and increase the probability of getting the optimal solution, but there may still exist allocation error in AMS[2]. The direct allocation (DA) approach[9] generates all the moments in AMS without allocation error, whereas more on-line computation time is needed to search the correct facet that is aligned with the DM, which might prohibit the real-time application of the algorithm. Besides DA, several l_1-norm error minimization objective optimization methods can be converted to normal linear programming (LP) models[3] and be solved using simplex or interior-point method. These approaches are guaranteed to get an optimal solution in a finite period of time, but the allocation time is still three times or more than that of MPIR. The fast control allocation approach[7] using spherical coordinates behaves wonderfully on on-line computations, but a large amount of memory space is needed to save the look-up table in off-line computations. Durham[10] provided a computationally efficient method to get the optimal solution in more than 90% of examined cases, and the computation time varies linearly with the increasing number of controls. However, the method is hard to reproduce because of the lack of details[3]. It is hence crucial to find such an algorithm that performs superiorly in on-line computations without allocation error, and uses limited off-line computations and acceptable memory space.

In this paper, we focus on DA approach and introduce a method for greatly reducing the computation time of the algorithm. A new way of finding the vertices of AMS is introduced, which decreases the off-line computations when the number of actuators is less than 10. And a modified method of searching the facet that is aligned with DM is proposed to decrease the on-line computations, which checks only the facets that lie around DM and performs very well whatever the number of controls varies. The actuators of F-18 high angle-of-attack/alpha research vehicle (HARV) and a model of a spacecraft with 8 thrusters are used to illustrate the improvement of the method.

2. Problem Statement

Control allocation considers the problem of distributing the control requirement to redundant actuators in the closed loop of the control system. Fig.1 shows such a system with a module of control allocation. The control allocation is separated from the module of control strategy and deals with the problem of generating the commands of the redundant actuators to meet the desired moment $\boldsymbol{a}_\mathrm{d}$, which is calculated by the control strategy according to the current state and the expected state of the system. The separation simplifies the design of the control strategy since it no longer considers the constraints of the actuators. It also makes the control strategy and the control allocation algorithm applicable for different types of control systems. Additionally, the system becomes more robust since the separation minimizes the change of the system when the module of fault detection and diagnosis indicates that some actuators have failed to work (only the initial conditions of control allocation about the actuators should be updated).

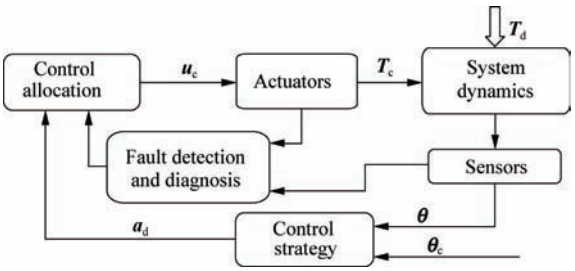

Fig.1 A closed-loop control system with the module of control allocation.

In Fig.1, $\boldsymbol{\theta}_\mathrm{c}$ is expected state vector, $\boldsymbol{\theta}$ measured state vector, $\boldsymbol{a}_\mathrm{d}$ desired moment vector, $\boldsymbol{u}_\mathrm{c}$ command vector of actuators, $\boldsymbol{T}_\mathrm{c}$ torque output vector of actuators, and $\boldsymbol{T}_\mathrm{d}$ disturbance torque vector.

Here we discuss the attitude control of a spacecraft using thrusters, where the degree of freedom (DOF) of the control requirement is 3. Assume there are m thrusters actuated to generate moment, the linearized transformation model is given by

$$\boldsymbol{a}_\mathrm{d} = \boldsymbol{A}\boldsymbol{u} \tag{1}$$

where $\boldsymbol{u} \in \mathbf{R}^m$ is the control vector, $\boldsymbol{A}$ the $3\times m$ control

matrix of effectiveness. For redundant control allocation problems, $m>3$. All the thrusters are assumed to have both minimum and maximum position limits, which span a convex hull of control space in m-dimensional (m-D) space:

$$\boldsymbol{\Omega} = \{\boldsymbol{u} \mid u_{i,\min} \le u_i \le u_{i,\max}, \boldsymbol{u} \in \mathbf{R}^m\} \subset \mathbf{R}^m \quad (2)$$

where $\boldsymbol{\Omega}$ is denoted as the the control subset, u_i the ith element of $\boldsymbol{u}$, and $u_{i,\min}$ and $u_{i,\max}$ are the minimum and maximum output limits of u_i respectively. A subset in 3-dimensional (3-D) space, denoted as $\boldsymbol{\Phi}$, is formed when the controls generate moments through the mapping matrix $\boldsymbol{A}$:

$$\boldsymbol{\Phi} = \{\boldsymbol{a}_{\mathrm{d}} \mid \boldsymbol{a}_{\mathrm{d}} = \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u} \in \boldsymbol{\Omega}\} \subset \mathbf{R}^3 \quad (3)$$

where $\boldsymbol{\Phi}$ is called the AMS consisting of all the moments that the actuators could generate in moment space. $\boldsymbol{\Phi}$ is also a convex hull due to the fact that linear transformation keeps convexity.

With these definitions, the problem of control allocation can now be stated as follows: given a DM, find a control vector $\boldsymbol{u}$ in $\boldsymbol{\Omega}$ that satisfies Eq.(1). We will also assume that each 3×3 sub-matrix of $\boldsymbol{A}$ is full rank, which ensures that each point of the boundary of $\boldsymbol{\Phi}$ is mapped by a unique control vector $\boldsymbol{u}$ in $\boldsymbol{\Omega}$. The assumption should be satisfied when designing the configuration of the actuators.

3. AMS and DA

In this section, we give a brief description of the DA method, and introduce a new approach of constructing AMS which performs superiorly to the one used in DA method when the number of actuators is less than 10.

3.1. Properties of AMS

AMS is constructed by finding its boundary which is the image of the facets of the control space. There are $2^{m-2}m!/[2!(m-2)!]$ facets in the control space, but only m^2-m facets of them map to be the boundary of the AMS under the assumption proposed in Section 2. Each facet of AMS is a parallelogram with four vertices and four edges. A vertex of AMS is mapped by a vertex of $\boldsymbol{\Omega}$ whose elements are totally at their position limits. An edge of AMS is mapped by an edge of $\boldsymbol{\Omega}$, which connects two vertices with $m-1$ elements at the same position limits, and the remaining one element varies. A facet of AMS is mapped by a facet of $\boldsymbol{\Omega}$ with $m-2$ elements at the same position limits and the remaining two vary. AMS has m^2-m+2 vertices, $2(m^2-m)$ edges and m^2-m facets if each 3×3 sub-matrix of $\boldsymbol{A}$ is full rank.

3.2. Computation of AMS

The facets and the edges of AMS are completely constructed by getting the connecting information of the vertices of AMS, so the vertices of AMS must be firstly constructed. Now we present two ways of finding the vertices. Before that a preparation theorem is given to confirm the extent of the vertices.

(1) A preparation theorem

It is known from Ref.[1] that the boundary of AMS, denoted as $\partial(\boldsymbol{\Phi})$, is the image of the boundary of $\boldsymbol{\Omega}$, denoted as $\partial(\boldsymbol{\Omega})$. Each component of $\partial(\boldsymbol{\Omega})$ has some or all of its controls saturated. Now we define $\overline{\boldsymbol{u}}$ whose controls are all saturated. Apparently, $\overline{\boldsymbol{u}} \in \partial(\boldsymbol{\Omega})$. Then the theorem is stated as follows:

Theorem 1 All the vertices of $\partial(\boldsymbol{\Phi})$ are the subset of $\boldsymbol{A}\overline{\boldsymbol{u}}$.

Proof First we should know that all the vertices of $\partial(\boldsymbol{\Phi})$ are mapped by $\overline{\boldsymbol{u}}$. It is a standard result on convex polytopes[11] and can also be found in Ref.[8].

Then it should just be proved that the vertices of $\partial(\boldsymbol{\Phi})$ are the subset of $\boldsymbol{A}\overline{\boldsymbol{u}}$, which is equivalent to finding a control vector in $\overline{\boldsymbol{u}}$ that does not map to $\partial(\boldsymbol{\Phi})$. See that the null space of $\boldsymbol{A}$, denoted as $\boldsymbol{N}(\boldsymbol{A})$, is defined as

$$\boldsymbol{N}(\boldsymbol{A}) = \{\boldsymbol{\xi} \mid \boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{0}, \boldsymbol{\xi} \in \mathbf{R}^m\} \quad (4)$$

Now find a basis vector, $\boldsymbol{\xi}_k$, of $\boldsymbol{N}(\boldsymbol{A})$, so

$$\boldsymbol{A}\boldsymbol{\xi}_k = \boldsymbol{0} \quad (5)$$

then a control vector $\boldsymbol{u}_k$ can be found in $\overline{\boldsymbol{u}}$ according to the elements of $\boldsymbol{\xi}_k$:

$$u_{k,i} = \begin{cases} u_{k,i,\max} & \xi_{k,i} > 0 \\ u_{k,i,\min} & \xi_{k,i} < 0 \\ u_{k,i,\max} \text{ or } u_{k,i,\min} & \xi_{k,i} = 0 \end{cases} \quad (i = 1, 2, \cdots, m) \quad (6)$$

Consider the moment vector $\boldsymbol{m}_k$ generated by $\boldsymbol{u}_k$:

$$\boldsymbol{A}\boldsymbol{u}_k = \boldsymbol{m}_k \quad (7)$$

See that there exist some other control vectors generating the same $\boldsymbol{m}_k$:

$$\boldsymbol{A}\boldsymbol{u}_k' = \boldsymbol{A}(\boldsymbol{u}_k + s\boldsymbol{\xi}_k) = \boldsymbol{A}\boldsymbol{u}_k + s\boldsymbol{A}\boldsymbol{\xi}_k = \boldsymbol{m}_k + \boldsymbol{0} = \boldsymbol{m}_k \quad (8)$$

then there must exist $s<0$ such that $\boldsymbol{u}_k'$ is an unsaturated control vector which can definitely not map to the vertex of $\partial(\boldsymbol{\Phi})$. Therefore, $\boldsymbol{u}_k$ cannot map to a vertex either, since it maps to $\boldsymbol{m}_k$ as well. Overall, all the vertices of $\partial(\boldsymbol{\Phi})$ are the subset of $\boldsymbol{A}\overline{\boldsymbol{u}}$.

The theorem indicates that, the matrix $\boldsymbol{A}$ determines the classification of $\boldsymbol{A}\overline{\boldsymbol{u}}$. A control vector $\boldsymbol{u}$ in the control space could be written as follows:

$$\boldsymbol{u} = \boldsymbol{x} + \boldsymbol{y} \quad (\boldsymbol{x} \perp \boldsymbol{y}, \boldsymbol{x} \in \boldsymbol{R}(\boldsymbol{A}), \boldsymbol{y} \in \boldsymbol{N}(\boldsymbol{A}), \boldsymbol{u} \in \boldsymbol{\Omega} \subset \mathbf{R}^m) \quad (9)$$

where $\boldsymbol{R}(\boldsymbol{A})$ is the row space of $\boldsymbol{A}$, which is the orthogonal complement of $\boldsymbol{N}(\boldsymbol{A})$ in $\mathbf{R}^m$ (namely, $\boldsymbol{N}(\boldsymbol{A}) \cap \boldsymbol{R}(\boldsymbol{A}) = \{\boldsymbol{0}\}$, and $\dim(\mathbf{R}^m) = \dim(\boldsymbol{N}(\boldsymbol{A})) + \dim(\boldsymbol{R}(\boldsymbol{A}))$, where $\dim(\sum)$ means the dimension of $\sum$[12]. So the elements

of $\bar{\boldsymbol{u}}$ whose projection on $\boldsymbol{N}(\boldsymbol{A})$ is $\boldsymbol{0}$ will map to be the vertices of $\partial(\boldsymbol{\Phi})$ (otherwise, $\boldsymbol{A}\bar{\boldsymbol{u}}$ can also be expressed by other unsaturated control vectors since $\boldsymbol{y}\neq\boldsymbol{0}$ and will not be the vertex of $\partial(\boldsymbol{\Phi})$). One way of finding such elements is based on the row space of $\boldsymbol{A}$, which means finding these elements directly; the other is based on the null space of $\boldsymbol{A}$, i.e., deleting the elements that map to the interior of $\boldsymbol{\Phi}$ and keeping the elements that map to the vertices. The vertices of $\partial(\boldsymbol{\Phi})$ are formed by multiplying the matrix $\boldsymbol{A}$ after finding these elements.

(2) Finding the vertices based on the row space of $\boldsymbol{A}$

This method was stated in Ref.[13] and is a search of the vertices of $\boldsymbol{\Omega}$ that map to $\partial(\boldsymbol{\Phi})$ using the row space of $\boldsymbol{A}$. First, take two columns of $\boldsymbol{A}$, $\boldsymbol{A}_i=[A_{i1}\quad A_{i2}\quad A_{i3}]^{\mathrm{T}}$ and $\boldsymbol{A}_j=[A_{j1}\quad A_{j2}\quad A_{j3}]^{\mathrm{T}}$, associated with two controls of $\boldsymbol{u}$, u_i and u_j, and two parallel facets of $\partial(\boldsymbol{\Phi})$. Get direction $\boldsymbol{t}=[t_1\quad t_2\quad t_3]^{\mathrm{T}}$ that is perpendicular to the two facets and satisfies

$$\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}_i=0,\ \boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}_j=0 \tag{10}$$

$\boldsymbol{t}$ can be solved by dividing the equation into two parts:

$$\begin{bmatrix}A_{i1} & A_{j1}\\ A_{i2} & A_{j2}\end{bmatrix}\begin{bmatrix}t_1\\ t_2\end{bmatrix}+t_3\begin{bmatrix}A_{i3}\\ A_{j3}\end{bmatrix}=\boldsymbol{0} \tag{11}$$

where t_3 is typically set to 1 and Eq.(11) has a solution if the 2×2 sub-matrix is invertible. Otherwise, the other 2×2 sub-matrices of $[\boldsymbol{A}_i\quad \boldsymbol{A}_j]$ can substitute it to get the solution. At least one of the sub-matrices will be invertible if all the 3×3 sub-matrices of $\boldsymbol{A}$ have rank 3.

Given that $\boldsymbol{t}$ is known, the vector $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}$ is used to determine the vertices and the facets of AMS. The vector $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}$ would have two zero elements, some positive elements and the remaining negative. The first four vertices will shape the highest facet in the direction $\boldsymbol{t}$, corresponding to the control vectors that have controls which correspond to positive elements of $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}$ set to maximum values, and controls which correspond to negative elements of $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}$ set to minimum values and u_i and u_j set to their combinations of possible position limits. The following four vertices will shape the lowest facet in the opposite direction of $\boldsymbol{t}$, corresponding to the control vectors that have controls which correspond to positive elements of $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}$ set to minimum values, and controls which correspond to negative elements of $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}$ set to maximum values and u_i and u_j set to their combinations of possible position limits. For example: if $\boldsymbol{u}_{\max}=[1\quad 1\quad 1\quad 1\quad 1]^{\mathrm{T}}$, $\boldsymbol{u}_{\min}=[-1\quad -1\quad -1\quad -1\quad -1]^{\mathrm{T}}$, and $\boldsymbol{t}^{\mathrm{T}}\boldsymbol{A}=[1\quad 0\quad -1\quad 0\quad -2]^{\mathrm{T}}$ (i=2 and j=4), then the four vertices of the highest facet perpendicular to the direction $\boldsymbol{t}$ would be $\boldsymbol{u}_1=[1\quad -1\quad -1\quad -1\quad -1]^{\mathrm{T}}$, $\boldsymbol{u}_2=[1\quad -1\quad -1\quad 1\quad -1]^{\mathrm{T}}$, $\boldsymbol{u}_3=[1\quad 1\quad -1\quad -1\quad -1]^{\mathrm{T}}$ and $\boldsymbol{u}_4=[1\quad 1\quad -1\quad 1\quad -1]^{\mathrm{T}}$, and the four vertices of the lowest facet perpendicular to the opposite direction of $\boldsymbol{t}$ would be $\boldsymbol{u}_5=[-1\quad -1\quad 1\quad -1\quad 1]^{\mathrm{T}}$, $\boldsymbol{u}_6=[-1\quad -1\quad 1\quad 1\quad 1]^{\mathrm{T}}$, $\boldsymbol{u}_7=[-1\quad 1\quad 1\quad -1\quad 1]^{\mathrm{T}}$ and $\boldsymbol{u}_8=[-1\quad 1\quad 1\quad 1\quad 1]^{\mathrm{T}}$.

All the vertices of $\partial(\boldsymbol{\Omega})$ that map to be the vertices of $\partial(\boldsymbol{\Phi})$ will be found by considering all the possible pairs of taking any two columns of $\boldsymbol{A}$, and the facets of $\partial(\boldsymbol{\Omega})$ that map to $\partial(\boldsymbol{\Phi})$ will be obtained simultaneously. The vertices of $\partial(\boldsymbol{\Phi})$ are then obtained by multiplying the matrix $\boldsymbol{A}$. These vertices of $\partial(\boldsymbol{\Omega})$ and $\partial(\boldsymbol{\Phi})$ should be saved for on-line computations. These vertices, however, will be saved repeatedly since each vertex is shared by two or more facets. Therefore, a numbering system is introduced to number the vertices for nonrecurring storage.

(3) Numbering the vertices

The method presented in Ref.[10] is simple and suitable for numbering the vertices. With this method, all the facets of AMS would be constructed in a facet table which could be found by the sequence numbers of the vertices. There are $4(m^2-m)$ sequence numbers of m^2-m facets and m^2-m+2 vertices to be saved.

(4) Finding the vertices based on the null space of $\boldsymbol{A}$

In this section, we present a new way of finding the vertices utilizing the null space of $\boldsymbol{A}$. The number of the elements of $\bar{\boldsymbol{u}}$ is 2^m, due to the fact that m controls have both minimum and maximum position limits. These elements correspond to the 2^m vertices of $\boldsymbol{\Omega}$ in control space; m^2-m+2 map to be the vertices of $\partial(\boldsymbol{\Phi})$ and the remaining to the interior of $\boldsymbol{\Phi}$. The new approach deletes all the vertices of $\boldsymbol{\Omega}$ that map to the interior of $\boldsymbol{\Phi}$, and keeps the vertices of $\boldsymbol{\Omega}$ that map to the vertices of $\boldsymbol{\Phi}$. Consider a basis vector of $\boldsymbol{N}(\boldsymbol{A})$ that satisfies

$$\boldsymbol{A}\boldsymbol{\xi}=\boldsymbol{0} \tag{12}$$

If all the elements of $\bar{\boldsymbol{u}}$ like the one shown in Eq.(6) are found, then all the elements that map to the interior of $\boldsymbol{\Phi}$ will be found and deleted. Select a 3×3 sub-matrix of $\boldsymbol{A}$ arbitrarily and denote their corresponding columns as $\boldsymbol{A}_i$, $\boldsymbol{A}_j$ and $\boldsymbol{A}_k$. Evaluate $m-3$ column vectors as follows:

$$\boldsymbol{x}_1=\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix},\ \boldsymbol{x}_2=\begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix},\ \cdots,\ \boldsymbol{x}_{m-3}=\begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix} \tag{13}$$

which correspond to $m-3$ elements of the basis vectors of $\boldsymbol{\xi}$ except its ith, jth and kth elements. Take each of the vectors into Eq.(12) and the ith, jth and kth elements of $\boldsymbol{\xi}_l$ (l=1, 2, ⋯, $m-3$) can be solved if the 3×3 sub-matrix of $\boldsymbol{A}$ is invertible:

$$\begin{bmatrix}\xi_{l,i}\\ \xi_{l,j}\\ \xi_{l,k}\end{bmatrix}=[\boldsymbol{A}_i\quad \boldsymbol{A}_j\quad \boldsymbol{A}_k]^{-1}\tilde{\boldsymbol{A}}\boldsymbol{x}_l\qquad (l=1,2,\cdots,m-3) \tag{14}$$

where $\tilde{\boldsymbol{A}}$ is the sub-matrix of $\boldsymbol{A}$ removing its ith, jth and kth columns. The vertices $\boldsymbol{u}_l=[u_{l,1}\quad u_{l,2}\quad \cdots\quad u_{l,m}]^{\mathrm{T}}$ (l=1,2, ⋯,$m-3$) that map to the interior of $\boldsymbol{\Phi}$ can then be found based on $\boldsymbol{\xi}_l$:

$$u_{l,i}=\begin{cases}u_{l,i,\max} & \xi_{l,i}>0\\ u_{l,i,\min} & \xi_{l,i}<0\\ u_{l,i,\max}\ \text{or}\ u_{l,i,\min} & \xi_{l,i}=0\end{cases}\qquad (i=1,2,\cdots,m)\quad (15)$$

or

$$u_{l,i}=\begin{cases}u_{l,i,\min} & \xi_{l,i}>0\\ u_{l,i,\max} & \xi_{l,i}<0\\ u_{l,i,\max}\ \text{or}\ u_{l,i,\min} & \xi_{l,i}=0\end{cases}\qquad (i=1,2,\cdots,m)\quad (16)$$

There are $m(m-1)(m-2)/6!$ combinations of groupings when taking 3 columns from m columns, and $m-3$ basis vectors in each combination. However, some of the vertices may appear repeatedly in different combinations, which would be deleted once it appears.

All of the vertices that map to the interior of $\boldsymbol{\Phi}$ will be deleted using this method, keeping the vertices that map to the vertices of $\boldsymbol{\Phi}$. The vertices of $\boldsymbol{\Phi}$ are then formed by multiplying the matrix $\boldsymbol{A}$. The facets of $\boldsymbol{\Phi}$ are easy to construct when knowing the vertices of $\boldsymbol{\Phi}$. Take two elements of $\boldsymbol{u}$ and make them vary, and find the vertices that have the same values for the remaining elements. Each four vertices that have the same two elements vary and the remaining at the same position limits compose a facet. Two parallel facets will be found in each circulation decided by Eqs.(15)-(16).

(5) Evaluation of the two methods

The two ways of constructing AMS may be suitable for different numbers of actuators. Fig.2 shows the comparison of time consumed to the construction of AMS. We use the configuration of F-18 HARV to test the methods whose matrix of control effectiveness and limits are given by

$$\boldsymbol{A}=\begin{bmatrix}-4.382 & 4.382 & -5.841 & 5.481 & 1.674 & -6.280 & 6.280 & 2.920 & 0.001 & 1.000\\ -53.30 & -53.30 & -6.486 & -6.486 & 0 & 6.234 & 6.234 & 0.001 & 35.53 & 0.001\\ 1.100 & -1.100 & 0.391\,1 & -0.391\,1 & -7.482 & 0 & 0 & 0.030 & 0.001 & 14.85\end{bmatrix}\times 10^{-2}\quad (17)$$

$$\boldsymbol{u}_{\min}=[-4.189\ \ -4.189\ \ -5.236\ \ -5.236\ \ -5.236\ \ -1.396\ \ -1.396\ \ -5.236\ \ -5.236\ \ -5.236]\times 10^{-1}\quad 18)$$

$$\boldsymbol{u}_{\max}=[1.833\ \ 1.833\ \ 5.236\ \ 5.236\ \ 5.236\ \ 7.854\ \ 7.854\ \ 5.236\ \ 5.236\ \ 5.236]\times 10^{-1}\quad (19)$$

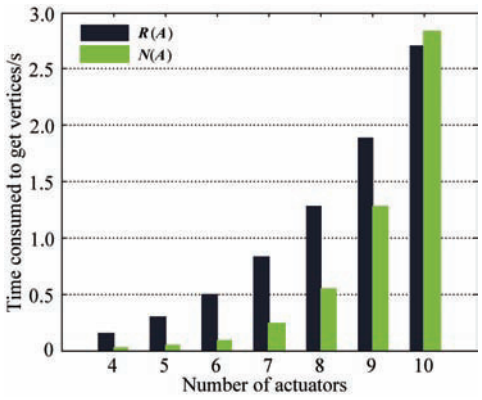

Fig.2 Time comparison of off-line constructing AMS based on $\boldsymbol{R}(\boldsymbol{A})$ and $\boldsymbol{N}(\boldsymbol{A})$.

Note that as the number of actuators increases, the time consumed to construct AMS increases simultaneously when using either of the methods. However, the method based on the null space of $\boldsymbol{A}$ saves at least 32.22% (1.281 s compared with 1.890 s when m=9) of off-line computation time when the number of actuators is less than 10. When $4<m<6$, the number of the vertices of $\boldsymbol{\Omega}$ to be deleted is less than that to be saved; when $6\le m\le 9$, the number of the vertices of $\boldsymbol{\Omega}$ to be deleted is more than or equal to that of the vertices of $\boldsymbol{\Omega}$ to be saved, but the time consumed to find a vertex is longer than that consumed to delete a vertex, so the method based on the null space of $\boldsymbol{A}$ is still superior. The method based on the row space of $\boldsymbol{A}$ is suitable for constructing the AMS when $m\ge 10$.

After the construction of AMS, a 3-D view of the AMS is shown in Fig.3. The set is delimited by 90 facets, 180 edges and 92 vertices. In Fig.3, $T_{c,x}$, $T_{c,y}$ and $T_{c,z}$ represent the x, y and z axes of 3-D torque space.

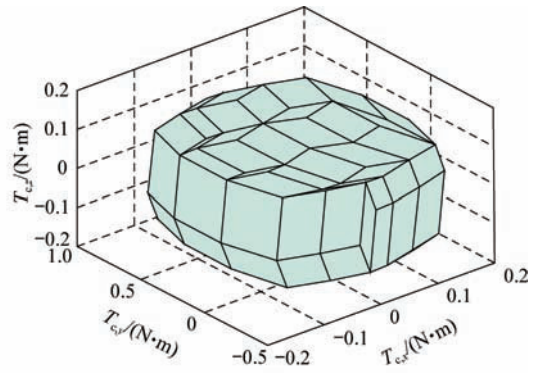

Fig.3 AMS for F-18 HARV.

3.3. Computation of control input

The computation of control input involves three vertices of a facet. Fig.4 shows the principium of the computation.

A facet could be described by the linear combination of three of its four vertices:

$$\begin{cases}a\boldsymbol{a}_{\mathrm{d}}=\boldsymbol{a}_{\mathrm{d},i}+b(\boldsymbol{a}_{\mathrm{d},j}-\boldsymbol{a}_{\mathrm{d},i})+c(\boldsymbol{a}_{\mathrm{d},k}-\boldsymbol{a}_{\mathrm{d},i})\\ a>0,\ 0\le b\le 1,\ 0\le c\le 1\end{cases}\qquad (20)$$

where $\boldsymbol{a}_{\mathrm{d},i}$ is a basis vertex of the facet with its two adjacent vertices, namely, $\boldsymbol{a}_{\mathrm{d},j}$ and $\boldsymbol{a}_{\mathrm{d},k}$. a is used to

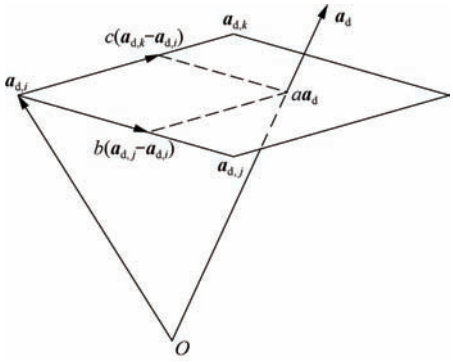

Fig.4 Sketch of computing of control input.

scale the DM $\boldsymbol{a}_d$ to intersect with the facet. a, b and c are solved as follows:

$$[a \quad b \quad c]^T=[\boldsymbol{a}_{d,i} \quad \boldsymbol{a}_{d,j}-\boldsymbol{a}_{d,j} \quad \boldsymbol{a}_{d,i}-\boldsymbol{a}_{d,k}]^{-1}\boldsymbol{a}_d \quad (21)$$

If there existed an intersection between the facet and the scaled DM, the three parameters would satisfy the condition shown in Eq.(20). The intersection represents the maximum attainable moment (MAM) that the actuators could generate in this direction. Assume that three vertices of $\boldsymbol{\Omega}$, denoted as $\boldsymbol{u}_i$, $\boldsymbol{u}_j$ and $\boldsymbol{u}_k$, generate $\boldsymbol{a}_{d,i}$, $\boldsymbol{a}_{d,j}$ and $\boldsymbol{a}_{d,k}$, separately. The control vector $\boldsymbol{u}_d$ that generates $\boldsymbol{a}_d$ is then computed as follows:

$$\begin{cases} \boldsymbol{u}_d^*=\boldsymbol{u}_i+b(\boldsymbol{u}_j-\boldsymbol{u}_i)+c(\boldsymbol{u}_k-\boldsymbol{u}_i) \\ \boldsymbol{u}_d=\begin{cases} \boldsymbol{u}_d^*/a & a\geq 1 \\ \boldsymbol{u}_d^* & a<1 \end{cases} \end{cases} \quad (22)$$

where $\boldsymbol{u}_d^*$ represents the control vector that generates the MAM along $\boldsymbol{a}_d$. If $a\geq 1$, $\boldsymbol{a}_d$ is attainable, and $\boldsymbol{u}_d^*$ will be scaled to $\boldsymbol{u}_d$ to generate $\boldsymbol{a}_d$; if $a<1$, $\boldsymbol{a}_d$ is unattainable, and $\boldsymbol{u}_d$ will be equal to $\boldsymbol{u}_d^*$, generating the MAM along the direction.

3.4. DA method

In Durham's method[9] Eq. (21) is used as a test of whether the facet is intersected with DM. If $[a \quad b \quad c]$ satisfies the limits in Eq. (20), the facet intersects with DM, and the control input can be calculated using Eq.(22); otherwise, the algorithm continues to search for the correct facet. All the possible facets of AMS are checked sequentially and the correct facet will be found in a limited period of time. The procedure is referred to as the sequential search in Ref.[7]

The computations are split into off-line and on-line computations. Off-line computations construct the information of AMS, including the vertices and the facets of AMS, and the control vectors associated with the vertices of AMS. On-line computations consist of finding the correct facet that is aligned with DM, and computing the control input when given a DM. Now we attempt to make the strategy more efficient.

4. Modified DA (MDA) Algorithm

The output of DA algorithm keeps the direction of DM, and all the moments in $\boldsymbol{\Phi}$ can be generated without allocation error (the allocation subset is equal to $\boldsymbol{\Phi}$). If DM lies out of AMS, MAM along the direction of DM will be the output. The main idea of DA algorithm is to find the correct facet that is aligned with DM, and great on-line allocation time of the algorithm has been consumed to search the facet. However, all the possible facets are checked sequentially, making the search time vary in an unacceptable broad range. And in the worst case, the correct facet will finally be checked which terribly prolongs the time of on-line computations and may prohibit its use for real-time applications.

The modified method presented here is still split into off-line computations and on-line computations. Off-line computations construct the information of AMS, including the vertices and facets of AMS, as well as the normalized vertices of AMS which would be utilized in the modified method. To reduce the off-line computation time, the method based on the null space of $\boldsymbol{A}$ is used to find the vertices of $\boldsymbol{\Omega}$ that map to the vertices of AMS when the number of actuators is less than 10. Otherwise, the method based on the row space of $\boldsymbol{A}$ is used. On-line computations search the correct facet that is aligned with DM, and calculate the commands of the actuators. It is expected that the modified method would keep the properties of AMS-based approaches, and perform more efficiently than any existing allocation method.

4.1. Off-line computations: construction of AMS

The construction of AMS has been stated in Section 3.2. Additionally, the vertices of AMS need to be normalized and saved for the modified method:

$$\boldsymbol{a}_{d,v}^*=\frac{\boldsymbol{a}_{d,v}}{|\boldsymbol{a}_{d,v}|} \qquad (v=1,2,\cdots,m^2-m+2) \quad (23)$$

where $\boldsymbol{a}_{d,v}$ is the vth vertex of $\boldsymbol{\Phi}$, $\boldsymbol{a}_{d,v}^*$ the vth normalized vertex of $\boldsymbol{\Phi}$, and $|\boldsymbol{a}_{d,v}|$ the magnitude of the vertex. After the normalization, all the normalized vertices lie on the unit sphere so that the nearest vertices around a given DM could be found in on-line computations.

After off-line computations, the vertices of $\boldsymbol{\Phi}$, the normalized vertices of $\boldsymbol{\Phi}$, the facets of $\boldsymbol{\Phi}$ and the vertices of $\boldsymbol{\Omega}$ that map to be the vertices of $\boldsymbol{\Phi}$ will be saved for on-line computations.

4.2. On-line computations: search of intersectant facet and calculation of $\boldsymbol{u}_d$

In this section a more efficient method of calculating the controlling instruction will be presented. To improve the efficiency of searching the correct facet intersecting with the half line along DM, we attempt to use the information given by the unit direction of the moment (UDM). Consider that the DM could be written as follows:

$$\boldsymbol{a}_{\mathrm{d}}=\frac{\boldsymbol{a}_{\mathrm{d}}}{|\boldsymbol{a}_{\mathrm{d}}|}|\boldsymbol{a}_{\mathrm{d}}|=|\boldsymbol{a}_{\mathrm{d}}|\boldsymbol{a}_{\mathrm{d}}^{*} \tag{24}$$

where $|\boldsymbol{a}_{\mathrm{d}}|$ is the magnitude of the $\boldsymbol{a}_{\mathrm{d}}$, and $\boldsymbol{a}_{\mathrm{d}}^{*}$ UDM lying on the unit sphere with the normalized vertices of $\boldsymbol{\Phi}$.

After the normalization, the distances between the normalized vertices and $\boldsymbol{a}_{\mathrm{d}}^{*}$ could be calculated and sorted ascendingly. The sequence numbers of the nearest q normalized vertices against $\boldsymbol{a}_{\mathrm{d}}^{*}$ could be found by checking the smallest q distances:

$$\mathrm{Seq}_{v}=\min_{q}\{\| \boldsymbol{a}_{\mathrm{d},v}^{*}-\boldsymbol{a}_{\mathrm{d}}^{*} \|_{2}, v=1, 2, \cdots, m^{2}-m+2\} \tag{25}$$

where the function $\min_{q}\{\cdot\}$ means finding the nearest q normalized vertices against $\boldsymbol{a}_{\mathrm{d}}^{*}$. Generally, $q\geq 1$, owing to the fact that the intersecting facet may not correspond to the nearest normalized vertex against UDM if there exist some adjacent narrow facets. An example of the statement is given in Fig.5.

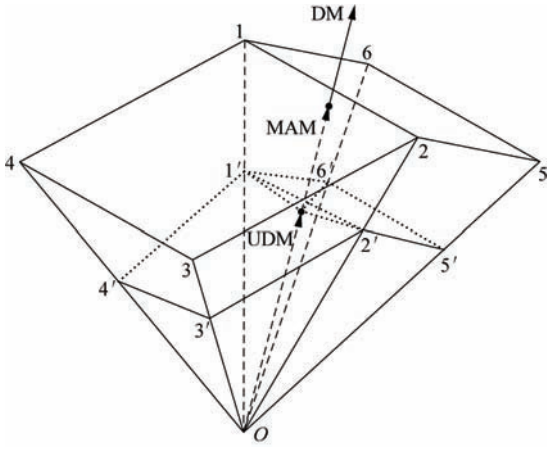

Fig.5 The closest vertex, but not within correct facet.

The vertices from 1 to 6 are some of the neighboring vertices of $\boldsymbol{\Phi}$, and their corresponding normalized vertices from 1′ to 6′ lie on the unit sphere centered at zero in 3-D moment space. The half-line along UDM intersects with the unit sphere, crossing to form the UDM, and intersects with the parallelogram formed by the vertices 1-4, crossing to form MAM along the direction (Note that the normalization has altered the shape of the facet, but it keeps the connecting relations of the four vertices of the facet). However, the normalized vertex nearest to UDM is vertex 6′ whose corresponding vertex, vertex 6, is not one of the vertices of the intersecting facet. The intersecting facet cannot be found if only the facets that share the vertex 6 are searched. Fortunately, the normalized vertex 2′ lies close enough to the intersection, and the intersecting facet will be found if the facets sharing vertex 2 are searched.

The evaluation of q must guarantee the inclusiveness of the intersecting facet when given any direction of DM, which depends on the number and the position limits of the actuators. Generally, q increases with the increasing number of actuators and narrow facets, associated with small ranges of the position limits of some actuators. But on the other hand, q cannot be too large to reduce the search time of the on-line computations, either.

After the nearest q normalized vertices against UDM are found, the facets that have the vertices corresponding to these normalized vertices could be found out from the look-up table of the facets created in off-line computations. The next procedure is to find the intersecting facet in these facets and then, generate the commands of the actuators as stated in Section 3.3.

4.3. Evaluation of MDA algorithm

The procedure of MDA algorithm is summarized as follows:

Step 1 Find the vertices of $\boldsymbol{\Omega}$ that map to the vertices of $\boldsymbol{\Phi}$, save the vertices of $\boldsymbol{\Phi}$ and their corresponding controls and number the vertices.

Step 2 Find the vertices of each facet of $\boldsymbol{\Phi}$ and save the sequence numbers of the vertices of each facet.

Step 3 Calculate the normalized vertices of $\boldsymbol{\Phi}$ and save them.

Step 4 Normalize DM to form UDM.

Step 5 Find the closest q normalized vertices of $\boldsymbol{\Phi}$ against UDM.

Step 6 Search the facets that include some of the vertices corresponding to q normalized vertices separately.

Step 7 Check the facets found in Step 6 sequentially to get the facet that intersects with the half-line along UDM.

Step 8 Compute the control vector that generates DM according to the corresponding three controls that generate three vertices of the intersecting facet.

The first 3 steps are calculated off-line and the information will not change until some actuators fail to be actuated. When the number of actuators is less than 10, the method based on the null space of $\boldsymbol{A}$ is used to calculate the vertices. Otherwise, the method based on the row space of $\boldsymbol{A}$ is used.

The latter steps of the method are performed on-line. Compared with the DA algorithm, the MDA algorithm searches the facets near the DM instead of $\partial(\boldsymbol{\Phi})$, which greatly reduces the number of facets to be checked and remarkably decreases the allocation time. A little larger amount of roomage needs to be added to save the normalized vertices of $\boldsymbol{\Phi}$, and an additional time is consumed to find the nearest normalized vertices against UDM, but it is worth paying such expense to get more real-time characteristic of the algorithm.

5. Numerical Examples

To prove the time reduction of MDA, some numeri-

cal simulation examples are given in this section. A spacecraft with a redundant configuration of 8 thrusters is introduced to illustrate the algorithm. The configuration of the thrusters is shown in Fig.6, which is located at the diagonal of the top and foot surface of the spacecraft. The range of the reaction force of each thruster is assumed to be [0, 1] N, and the matrix $\boldsymbol{A}$ is shown as follows:

$$\boldsymbol{A}=\begin{bmatrix} -0.185\,48 & -0.185\,48 & 0.185\,48 & 0.185\,48 & 0.312\,64 & 0.312\,64 & -0.312\,64 & -0.312\,64 \\ 0.312\,64 & -0.312\,64 & -0.312\,64 & 0.312\,64 & -0.185\,48 & 0.185\,48 & 0.185\,48 & -0.185\,48 \\ -0.101\,73 & 0.101\,73 & -0.101\,73 & 0.101\,73 & 0.101\,73 & -0.101\,73 & 0.101\,73 & -0.101\,73 \end{bmatrix} \tag{26}$$

The matrix satisfies the assumption that each 3×3 sub-matrix is invertible. The results of the simulation are obtained on a Pentium IV machine running at 2.4 GHz and using implementations of the algorithms as m-files in MATLAB 6.5. Timing results are obtained using the tic/toc commands and a rough comparison is provided between the MDA, the MPIR (Eq.(10) of Ref.[2], the DA (SIMP) (linear programming model of DA solved by simplex method[3]) method, and the SGO method[5].

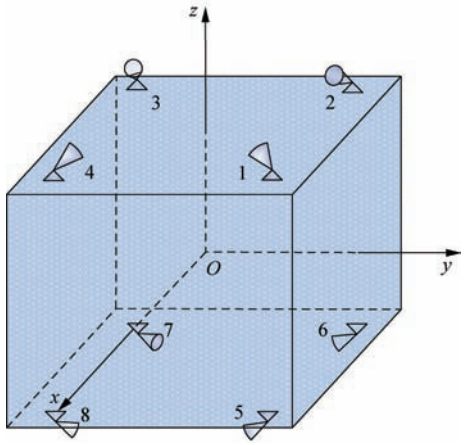

Fig.6 Configuration of thrusters.

5.1. Off-line computation results

In this part of simulation we test the time consumed in the off-line computations. The first way based on the row space of $\boldsymbol{A}$ spends 1.343 0 s to complete the construction, while the second way based on the null space of $\boldsymbol{A}$ spends 0.593 0 s. The time varies in a small range owing to the circumstance of hardware, but the first approach always spends more time. The approach based on the null space of $\boldsymbol{A}$ is hence preferred as for the construction of this model (m=8).

There are 58 vertices of $\boldsymbol{\Omega}$, 58 vertices of $\boldsymbol{\Phi}$, 58 normalized vertices of $\boldsymbol{\Phi}$ and 224 sequence numbers for the 56 facets of $\boldsymbol{\Phi}$ to be saved for the problem. Fig.7 shows the vertices of $\boldsymbol{\Phi}$, the facets of $\boldsymbol{\Phi}$ and their connectivity, and Fig.8 shows the normalized vertices of $\boldsymbol{\Phi}$.

5.2. On-line computation results

In this part of simulation we test the on-line computation of the algorithm. q is evaluated to be 3 for the problem, and 1 000 random desired moments are given to test the statistical characteristics of the four algorithms. Fig.9 shows a histogram of the number of facets to be checked before the correct facet is found. 86.8% of the samples find the correct facet after checking the first 4 facets, which means that the vertex

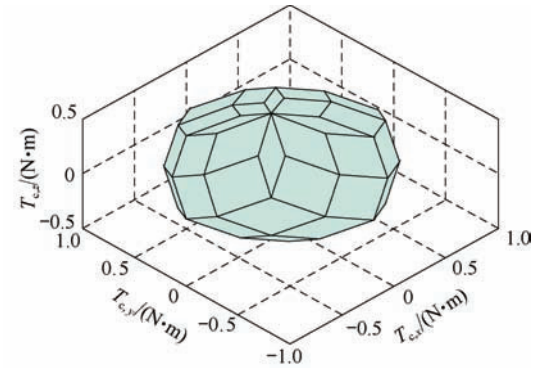

Fig.7 Vertices of $\boldsymbol{\Phi}$ and their connectivity.

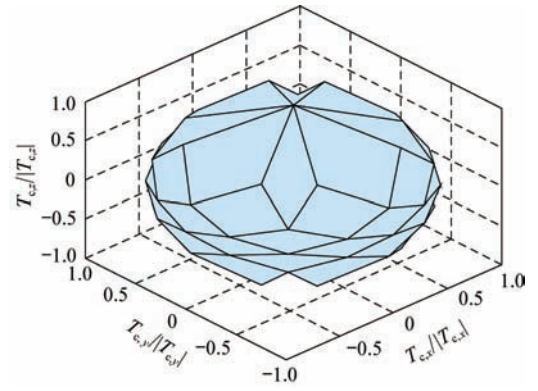

Fig.8 Normalized vertices of $\boldsymbol{\Phi}$ and their connectivity.

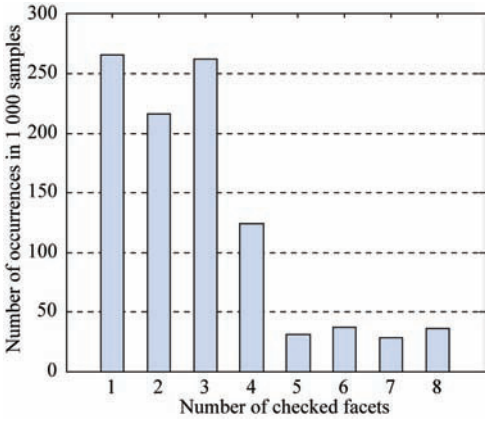

Fig.9 Histogram of checked facets.

nearest to the intersection between the half-line along DM and $\partial(\boldsymbol{\Phi})$ is most likely to be the component of the intersecting facet. No more than 8 facets are checked before the correct one is found. However, there are still 13.2% of the samples whose number of checking the facets are more than 4, due to the fact that two vertices of AMS are shared by 8 facets (see Figs.7-8), increasing the number of checked facets before the correct one is found.

Table 1 shows the on-line computation time of the modified algorithm during the test, compared with the most efficient MPIR[2], an LP modeled DA solved by simplex method (DA (SIMP))[3] and the SGO method[5]. The bisecting edge searching method[10] is out of the simulations because the same difficulty as encountered by Bodson[3] exists during the reproduction. The minimum calculating time appears when the correct facet is firstly checked, which might happen in both MDA and DA (SIMP), while MPIR gets the instruction of the controls with their constraints in the first loop. The maximum calculating time of DA (SIMP) appears when the correct facet is finally checked; while MPIR gets the solution after the maximum of 8 loops, which is the number of actuators. SGO runs a fixed number of eight iterations to get the commands, which makes its allocation time just vary in a very narrow range. The average time of MDA reduces by 49.40% compared with MPIR, 89.81% compared with DA (SIMP), and 64.29% compared with SGO. Additionally, MDA keeps the properties of AMS-based methods (direction preservation, without allocation errors in AMS), whereas MPIR and SGO may have allocation errors occasionally in AMS, especially when DM lies towards the boundary of AMS. However, the maximum allocation time of MDA is still 2.36 times longer than SGO. The comparison of the properties between these methods indicates that MDA is capable of being an alternative to other efficient allocation algorithms. But on the other hand, readers should apply MDA cautiously with the consideration of off-line computations, because MDA can definitely not achieve such predominant properties of on-line computations without the data being precomputed in off-line computations.

Table 1 Time comparison of on-line computations between MDA, MPIR, DA (SIMP) and SGO

Algorithm	Minimum time/ms	Maximum time/ms	Average time/ms
MDA	<1	16.00	0.85
MPIR	<1	16.00	1.68
DA (SIMP)	<1	63.00	8.34
SGO	2.30	6.78	2.38

The configuration can generate any direction of desired moment even any two of the 8 thrusters fail. Future studies will give theoretical support for this result. Simulation results show that no more than 6 facets will be checked in any failed case. Table 2 depicts the on-line computation time of MDA, compared with MPIR, DA (SIMP) and SGO, when thrusters 2 and 6 fail. Note that MDA is still superior to the other three algorithms, but off-line computations should also be considered with caution for applications.

Table 2 Time comparison of on-line computations between MDA, MPIR, DA (SIMP) and SGO (Thrusters 2&6 failed)

Algorithm	Minimum time/ms	Maximum time/ms	Average time/ms
MDA	<1	16.00	0.73
MPIR	<1	16.00	1.17
DA (SIMP)	<1	46.00	6.95
SGO	1.82	5.80	1.98

Fig.10 shows an additional characteristic of the algorithms. We use the model of F-18 HARV to test the average time of on-line computations with an increasing number of actuators. Note that MDA makes the average time stay at nearly the same level with the increasing number of actuators, which is determined by the fact that only the facets located near DM are checked whatever number of actuators the configuration has. The three other algorithms use gradually increasing time to get the results when the number of actuators increases (when m=4, the execution time is negligible without iteration for SGO[5]). The results indicate that MDA may be more real-time in performance than the other three methods in control allocation problems with larger number of actuators where the time executed for off-line computations may be easily acceptable.

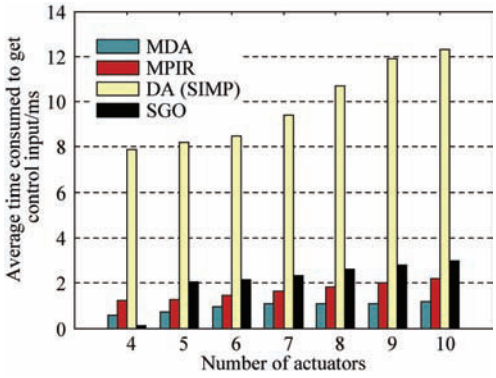

Fig.10 Average time of on-line computations.

6. Discussions

A control allocation algorithm based on the direct allocation is proposed to allocate the control requirements among redundant actuators. The algorithm modifies the direct allocation algorithm and is still split into off-line computations and on-line computations. During the off-line computation of constructing the attainable moment set, the method based on the null space of the effectiveness matrix proposed in this paper saves at least 32.22% of computation time com-

pared with the method based on the row space of the matrix, but it is only effective when the number of actuators is less than 10. Otherwise, the method based on the row space of the matrix is superior. On-line computations are remarkably reduced when only the facets around the desired moment are checked. When accounting for the spacecraft with 8 thrusters, numerical results show that the MDA algorithm saves 49.40% of allocation time compared with MPIR, 89.81% compared with DA (SIMP), and 64.29% compared with SGO, while the maximum allocation time is 2.36 times longer than SGO, with 2.76 times longer in failed mode.

Requirement of the memory space of the MDA algorithm is close to that of the direct allocation approach, while the on-line computation time performs much more superiorly whatever number of actuators the configuration has, but the results are totally obtained with sufficient off-line computations, which should be considered carefully for specific applications. In higher redundant control allocation problems, it would perform better since off-line computations may be more acceptable, as well as the on-line allocation time stays nearly at the same level. The approach provides a meaningful alternative to previously known efficient allocation strategies.

References

[1] Durham W. Constrained control allocation. Journal of Guidance, Control, and Dynamics 1993; 16(4): 717-725.
[2] Jin J. Modified pseudoinverse redistribution methods for redundant controls allocation. Journal of Guidance, Control, and Dynamics 2005; 28(5): 1076-1079.
[3] Bodson M. Evaluation of optimization methods for control allocation. Journal of Guidance, Control, and Dynamics 2002; 25(4): 703-711.
[4] Alwi H, Edwards C. Propulsion control of a large civil aircraft using on-line control allocation. 2009 American Control Conference. 2009; 4581-4586.
[5] Servidia P A, Peña R S. Spacecraft thruster control allocation problems. IEEE Transactions on Automatic Control 2005; 50(2): 245-249.
[6] Alwi H, Edwards C. Fault tolerant control using sliding modes with on-line control allocation. Automatica 2008; 44(7): 1859-1866.
[7] Petersen J A M, Bodson M. Fast control allocation using spherical coordinates. AIAA-1999-4215, 1999.
[8] Bordignon K A. Constrained control allocation for systems with redundant control effectors. PhD thesis, Virginia Polytechnic Institute and State University, 1996.
[9] Durham W. Constrained control allocation: three moment problem. Journal of Guidance, Control, and Dynamics 1994; 17(2): 330-336.
[10] Durham W. Computationally efficient control allocation. Journal of Guidance, Control, and Dynamics 2001; 24(3): 519-524.
[11] Ziegler G M. Lectures on polytopes (graduate texts in mathematics). New York: Springer-Verlag, 1995.
[12] Zhang X D. Matrix analysis and applications. Beijing: Tsinghua University Press, 2004: 592-604. [in Chinese]
[13] Durham W. Attainable moments for the constrained control allocation problem. Journal of Guidance, Control, and Dynamics 1994; 17(6): 1371-1373.

Biographies:

TANG Shengyong Born in 1985, he received his B.E. degree from Harbin Institute of Technology (HIT), Harbin, in 2007. From then on, he has been pursuing his Ph.D. study in science and technology of aeronautics and astronautics directly in HIT. His research interest is the modeling and simulation of the attitude control system of satellite.
E-mail: shermatang@gmail.com

ZHANG Shijie Born in 1979, he received his Ph.D. degree in science and technology of aeronautics and astronautics from Harbin Institute of Technology (HIT), Harbin, in 2005. Since graduation he has been working for the Research Center of Satellite Technology (RCST), HIT, concentrated on vision-based navigation, as well as the orbit, attitude control and system simulation of satellite.
E-mail: sjzhang@hit.edu.cn

ZHANG Yulin Born in 1958, he received B.S. and M.S. degrees from National University of Defense Technology (NUDT) in 1982 and 1984 respectively, and Ph.D. degree from Zhejiang University in 1988. After that, he visited the University of Waterloo (Canada) to continue his post doctoral research. He has worked as head of the department of Spacecraft Technology, NUDT, president of the Academy of Equipment Command & Technology, and commander of Jiuquan Satellite Lunch Center in succession since 1994. In July 2008, he was appointed as president of NUDT.
E-mail: y.l.zhang@263.net