JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 28, No. 6, November–December 2005

Optimization of Spacecraft Thruster Management Function

Finn Ankersen[*] and Shu-Fan Wu[†]
ESA, 2200 AG Noordwijk, The Netherlands
and
Alexander Aleshin,[‡] Alexander Vankov,[§] and Vladimir Volochinov[¶]
D-3-Group GmbH, D-12489 Berlin, Germany

Some new mathematical algorithms related to spacecraft thruster selection problems have been designed, implemented, and tested. The proposed approach is based on the concept of precomputing a set of optimal combinations of thrusters and storing the according data in the onboard computer (OBC) memory. During the nominal flight of a spacecraft, only a look-up procedure will be required to select, in the real-time mode, the right combination of not more than six thrusters (for a six-dimensional task) and their firing times. For "off-nominal" situations, when the spacecraft propulsion subsystem has to often operate at the edge of its capability and provide maximum linear and/or angular control accelerations, an additional algorithm has been proposed based on a modification of the parametric simplex method. Real-time software implementing the proposed algorithms requires from 0.05 ms to 0.21 ms to be executed on a SUN workstation with a 550-MHz UltraSparc processor. Regarding the size of OBC memory required for storing real-time computation data, only 52 KB are needed for a spacecraft with 20 thrusters. The user is provided with the flexibility to select the option, which better meets specific criteria and constraints such as execution time and accuracy of the optimal solution (at the expense of execution time or memory). In addition to C-code implementation, the software module (intended for real time) could be implemented as a standard SIMULINK library block and can be integrated with different simulators for test purposes. The software implemented in the project demonstrated effectiveness and robustness of the designed algorithms both in open- and closed-loop tests including various spacecraft maneuvers.

I. Introduction

THE purpose of onboard thruster management function (TMF) is to select specific thrusters and their firing intervals to realize force and torque command values coming from the control system of a spacecraft.

At the present time, there exist two main approaches for the thruster management problem: fixed and flexible. In the fixed case, all possible combinations of thrusters are fixed in a catalog and directly associated with predefined maneuvers and control modes of a spacecraft. Control systems of many spacecraft designed in the 1960s through to the early 1980s are based on this approach because it requires much less onboard computations. However, success in using this approach depends on many heuristic engineering considerations that are always spacecraft specific and varies significantly for different spacecraft and thruster configurations. Also, it typically results in increase of propellant consumption vs an optimal solution and degradation of other performance of the control system (accuracy of control, time to complete a maneuver, etc.). In the flexible case, special software is used onboard to solve the thruster selection problem (TSP) in real time to minimize propellant consumption or optimize some other cost function. As a well-known example of an early implementation of this kind, the digital autopilot of the space shuttle could be mentioned.[1] It is based on linear-programming (LP) task formulation and an algorithm based on the simplex method. Even earlier, in 1969, Crawford[2] proposed analytic investigation of corresponding LP optimization problem to search for appropriate thruster configurations.

There were several follow-up studies aimed at improving the performance and area of applicability of this approach,[3–5] and LP optimization remains accepted as an adequate formalism for TSP in the case of orbital and interplanetary spacecraft.

Since the invention of "basic" simplex method in the middle of the 20th century, there were many successful modifications to increase the computational efficiency of solving LP tasks. The most popular (in computational practice) is primal-dual interior point (feasible and infeasible) approach involving various quasi-Newton techniques.[6–8] But these researches were focused mainly on large-scale systems characterized by a very high dimension of variable and/or constraints. For the specific spacecraft TSP (where LP dimension is about 6×30), application of these methods would not result in a significant advantage. The main problems associated with this approach are significant amount of onboard computations and difficulties in validating the real-time convergence and output. Some of the potentially crucial consequences of using computational algorithms just mentioned (simplex method and its "successors") for TSP are as follows:

1) It might require too many iterations (unpredictable in advance) to find the optimum solution within a short time slot allocated for this procedure in the onboard computer; the latter typically has quite limited computational resources;

2) The output of standard algorithms often becomes unstable when input values are near or beyond physical capabilities (controllability envelope) of the system under control (see an example in Sec. IV.B).

Also, as it will be shown later in this paper, the original spacecraft TSP is more complex than the classic LP problem, for which the simplex method is used. In many cases it falls into the category of nonlinear or even integer-programming problems; all are characterized by high complexity. Therefore, a special analysis should be undertaken to correctly simplify the original task to obtain an approximate solution of the original problem on the base of the solution of LP problem (it is a special case of much

Received 14 June 2004; presented as Paper 2004-5133 at the AIAA Guidance, Navigation, and Control Conference, Providence, RI, 16–19 August 2004; revision received 27 May 2005; accepted for publication 21 March 2005.

[*]Senior GNC Engineer, P.O. Box 299, European Space Research and Technology Center; Finn.Ankersen@esa.int.

[†]Spacecraft Control Engineer, European Space Research and Technology Center; Shufan.Wu@esa.int. Senior Member AIAA.

[‡]Senior Software Engineer, Rudower Chaussee 29.

[§]Codirector and Project Manager, Rudower Chaussee 29; Alexander.Vankov@d3group.com.

[¶]Senior Research Scientist, Rudower Chaussee 29.

more general approach accepted in mathematical programming theory[9,10]).

As a baseline, the Automated Transfer Vehicle (ATV[11]) is considered. ATV is an unmanned servicing and logistics spacecraft to be used for the periodic resupply of the International Space Station (ISS).

The main goal of the present study is to provide a solution to the thruster management problem, which would take advantage of both fixed and flexible approaches and eliminate their drawbacks. The baseline is to precompute as much as possible in an off-line mode and then use these precomputed data to simplify real-time onboard computations. Such a concept was first suggested during an ESA research study on spacecraft rendezvous and docking, which took place in 1993–1995. Though the software implemented in that project[12] demonstrated outstanding real-time performance, it could only work with a certain type of control system and specific sampling time, as it was still based on heuristic assumptions.

One of the challenges of the present study, the results of which are reported in this paper, was to undertake a thorough mathematical research, resulting in a set of algorithms that would take into account specific features of real-time control of space vehicles. As a result, a number of original algorithms has been designed and implemented. The core algorithm to be used onboard the spacecraft is not using any simplex procedure at all. It is based on extensive precomputations of optimal bases of auxiliary LP problem to be provided off-line before the mission. The results are then stored onboard along with special look-up tables that enable for a quick search of either optimal LP problem solution, or a good approximation of an optimal solution without massive onboard calculations. In some cases, a special iterative procedure based on parametric-simplex[13] is invoked, which guarantees nondegradation of the solution being found at every next step. This is especially important for the hard-real-time mode when computations could be interrupted at the end of the time slot allocated for TMF.

A brief summary of the theoretical approach and results is presented in Sec. II. Based on these results, a dedicated software package was implemented that would help designers and analysts of spacecraft control systems to build, evaluate, and validate thruster management function. Its structure is described in Sec. III. Section IV summarizes the test results and performance achieved.

II. Summary of Theoretical Results

A. Assumptions and Nomenclature

A principal assumption of the present study is that TMF is separated from the spacecraft control system, as it takes place with the ATV spacecraft, for example. The control system calculates the vector of required forces and torques to be applied to the spacecraft, the latter considered as a six-degree-of-freedom rigid body. The TMF, in turn, calculates firing times for each individual thruster to deliver the required forces and torques with minimum propellant consumption.

Two different cases and corresponding mathematical problems have been investigated. Each case corresponds to a different physical implementation of propulsion drive electronics (PDE) and thruster design.

In the first case, it is assumed that any thruster could be open for any time within the sampling period that exceeds a so-called minimum open thruster interval; the latter is subject to electromechanical constraints of PDE, thruster and combustion process. As an example, the propulsion subsystem to be used on the ATV spacecraft could be mentioned. There, attitude and orbit control thrusters could be open, with a high accuracy, for any interval that exceeds 25 ms. This case is referred to as the continuous case.

The second case implies that for any thruster there is a discrete series of open thrust intervals for which the thruster can be opened (within the control subsystem sampling period). It will be called hereinafter as a discrete or integer case (referring to integer programming as a related mathematical domain). The case when any thruster can be either open or closed within the whole sampling period belongs to "discrete" problem formulation.

The following technical parameters of a spacecraft propulsion subsystem have been considered as inputs for mathematical formulation of the TSP:

1) N is the total number of thrusters.

2) M is the dimension of control vector coming from guidance, navigation, and control (GNC) (for translational and rotational motion $M = 6$).

3) Here $\boldsymbol{u} = (u_1, u_2, \ldots, u_M) \in \mathbb{R}^M$ is the vector of input control coming from GNC, defined in the six-dimensional space.

4) The $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_N) \in \mathbb{R}^N$ is the sought-for vector of open thrusters intervals (the output of TMF module).

5) The $\tau^{\min} > 0$ is the minimal open thruster interval (generally speaking, this value can vary for different thrusters as it is possible in the software designed).

6) T is the GNC sampling period.

7) $\mathcal{A} = (\mathcal{A}_{mn})_{m=1:M,\, n=1:N}$ is the $M \times N$ matrix of the direction of angular and translational accelerations that should be produced by the propulsion subsystem of the spacecraft.

8) Here $e_n(\tau_n)$, $n = 1 : N$ is the function of the module of acceleration produced by the nth thruster (engine efficiency per open thruster interval).

9) $I_{\mathrm{sp}n}(\tau_n)$, $n = 1 : N$ is the specific impulse of the nth thruster.

10) The $\phi_n(\tau_n) = [e_n(\tau_n) \cdot \tau_n]/[I_{\mathrm{sp}n}(\tau_n) \cdot g]$ is the fuel consumption function of the nth thruster with respect to thruster time, where g is the acceleration of gravity.

B. Formulation of Original Mathematical Problem

Now, in the continuous case the thruster selection problem has been formulated as follows:

$$\sum_{n=1}^{N} \phi_n(\tau_n) \to \min_{\tau}, \text{ s.t. (subject to)}$$

$$\sum_{n=1}^{N} \mathcal{A}_{mn} e_n(\tau_n) \tau_n = u_m \qquad m = 1 : M$$

$$\text{either} \qquad \tau_n = 0, \qquad \text{or} \qquad \tau^{\min} \le \tau_n \le T \qquad n = 1 : N \tag{1}$$

In the discrete case the following thruster selection problem has been considered:

$$\sum_{n=1}^{N} \phi_n(\tau_n) \to \min_{\tau}, \text{ s.t.}$$

$$\sum_{n=1}^{N} \mathcal{A}_{mn} e_n(\tau_n) \tau_n = u_m \qquad m = 1 : M$$

$$\tau_n \in \left\{0, \tau_n^{(1)}, \tau_n^{(2)}, \ldots, \tau_n^{(D)}\right\}, \qquad \tau^{\min} = \tau_n^{(1)}, \qquad \tau_n^{(D)} = T \qquad n = 1 : N \tag{2}$$

Here the finite set of time values $\{\tau_n^{(d)}, d = 1 : D\}$ represents the discrete series of open thruster intervals that are admissible by PDE; we assume for simplicity that $\tau_n^{(d)} = d \cdot \tau^{\min}$.

Both problems (1) and (2) are difficult nonlinear ones, and the goal of this mathematical research is to elaborate effective computational algorithms to obtain (at least an approximate) solution to these problems. The following basic requirements have been formulated:

1) *Robustness:* That is, the algorithm should return reasonable results for a rather wide set of possible input vectors $\boldsymbol{u}$ coming from the control system, even in cases when either input vector lies beyond the controllability envelope (the set of control vectors that can be provided by any combination of thrusters working during a sampling period), or when the computational process should be interrupted forcedly because of the fixed time slot allotted by the onboard computer (OBC) to the TMF routine.

2) *OBC resource consumption:* That is, duration of onboard calculations should be small enough, for example, less than 25% of T, and necessary computational resources (memory, processor performance) should be acceptable for a state-of-the-art onboard computer.

C. Reduction of Original Thruster Selection Problem

For real thrusters both engine efficiency and fuel consumption functions $e_n(\tau)$ and $\phi_n(\tau)$, respectively, are nonlinear ones with respect to the variable τ. Fortunately, analysis of typical thrusters' technical parameters (e.g., for the ATV program[11]) showed that $\phi_n(\tau)$ can be well approximated by a linear function and product $e_n(\tau)\cdot\tau$ by a simple discontinuous piecewise linear function. Namely, deviation of $e_n(\tau_n)/[I_{\mathrm{sp}n}(\tau_n)\cdot g]$ from some constant d_n does not exceed a few percent (5–7%), and propellant consumption can be well approximated by a linear function $\phi_n \approx f_n\cdot\tau_n$. As to $e_n(\tau)\cdot\tau$, this function can be well approximated as follows (where $e_n^{\min} = e_n(\tau_n^{\min})$ and q_n are some small constants):

$$e_n(\tau)\cdot\tau \approx \begin{cases} 0, & \text{if} \quad \tau < \tau_n^{\min} \\ (1+q_n)\cdot e_n^{\min}\cdot\tau - q_n\cdot e_n^{\min}\cdot\tau^{\min}, & \text{if} \quad \tau \ge \tau_n^{\min} \end{cases}$$

Let us introduce change of variables τ_n to γ_n, and appropriate auxiliary parameters ($\gamma_n^{\min}$, $\gamma_n^{\max}$, and A_{mn}) as follows:

$$\gamma_n(\tau_n) = \begin{cases} 0, & \text{if} \quad \tau_n < \tau_n^{\min} \\ f_n\cdot\tau_n - \dfrac{q_n\cdot f_n\cdot\tau_n^{\min}}{(1+q_n)}, & \text{if} \quad \tau_n \ge \tau_n^{\min} \end{cases}$$

$$\gamma_n^{\min} = \frac{f_n\cdot\tau_n^{\min}}{(1+q_n)}, \qquad \gamma_n^{\max} = T\cdot f_n - \frac{q_n\cdot f_n\cdot\tau_n^{\min}}{(1+q_n)} \qquad n = 1:N$$

$$A_{mn} = \mathcal{A}_{mn}\cdot\frac{(1+q_n)\cdot e_n^{\min}}{f_n} \qquad m = 1:M, \qquad n = 1:N$$

Further reduction is based on the assumption that all thrusters are "equal" in the following sense: $(q_n\cdot f_n\cdot\tau_n^{\min})/(1+q_n) = k$, $n = 1:N$. As a result (omitting some set of equations transformations), we obtain the following reduction of problems (1) and (2), respectively:

$$\sum_{n=1}^{N}\gamma_n + k\sum_{n=1}^{N}\operatorname{sgn}(\gamma_n) \to \min_{\gamma}, \text{ s.t.}$$
$$\sum_{n=1}^{N} A_{mn}\gamma_n = u_m \qquad m = 1:M$$
$$\text{either} \quad \gamma_n = 0, \quad \text{or} \quad \gamma_n^{\min} \le \gamma_n \le \gamma_n^{\max} \qquad n = 1:N \tag{3}$$

and

$$\sum_{n=1}^{N}\gamma_n + k\sum_{n=1}^{N}\operatorname{sgn}(\gamma_n) \to \min_{\gamma}, \text{ s.t.}$$
$$\sum_{n=1}^{N} A_{mn}\gamma_n = u_m \qquad m = 1:M$$
$$\gamma_n = d_n\cdot\gamma_n^{\min}, \qquad d_n \in \{0, 1, \ldots, D_n\} \qquad n = 1:N \tag{4}$$

where new variables γ have dimensions of fuel consumption, for example, γ_n represents nth thruster consumption (minus some small fixed amount of fuel spent for thruster transient process), and we have one-to-one relations between initial variables τ_n (having dimensions of open thrust interval) and new variables γ_n. And $\operatorname{sgn}(\cdot)$ is a well-known sign function that takes value (-1) for negative argument, 0 for zero argument, and $(+1)$ for positive argument. Though problems (3) and (4) are simpler than the original ones (1) and (2), they are still nonlinear because of the discontinuous sign function in the goal criterion. Moreover, the continuous problem (3) has a nonconvex set of feasible solutions because of a specific form of the third constraint. To avoid these problems, at the final step of simplification we consider the following optimization problem that is close to both Eqs. (3) and (4):

$$\sum_{n=1}^{N}\gamma_n \to \min_{\gamma}, \text{ s.t.}$$
$$\sum_{n=1}^{N} A_{mn}\gamma_n = u_m \qquad m = 1:M$$
$$0 \le \gamma_n \le \gamma_n^{\max} \qquad n = 1:N \tag{5}$$

Problem (5) is a classic continuous LP problem with a so-called box constraint on its variables (the last group of inequalities).

A major part of the mathematical research in the project concerned elaboration of effective numerical routines to synthesize an optimal solution to the problem (5), when the right side of the equality constraint (vector $\boldsymbol{u}$) varies. The issue is that the well-known simplex method[13] with its modern modifications[6–8,13] and their reference implementations, for example, MATLAB® LINPROG,[7] or SIMPLX[14] routines, are not optimized for usage in real-time embedded systems in which computational resources are often limited. Namely, these routines can return an absolutely irrelevant set of variables [infeasible according to the constraints in Eq. (5)] that do not correspond to the input control vector $\boldsymbol{u}$ neither in length nor in direction in the following cases: 1) when a feasible solution of Eq. (5) does not exist at all (It is the case when vector $\boldsymbol{u}$ is too long; it might be so for a number of reasons, e.g., large state error could cause the commanded force to be bigger than the maximum force that could be produced by the thrusters.), and 2) when the optimal solution exists, but the simplex routine requires too many iterations and the computing time goes beyond the time slot allotted by OBC for TMF, and this routine is interrupted forcedly. It can be also shown that any effective algorithm to solve LP problem (5) enables acquisition of an approximate solution of continuous and discrete TSP variants in a uniform manner with explicit error estimation (including optimal value of the goal function). Not going into details, which lie beyond the main subject of this paper devoted to real-time TMF routine, it has been shown that if we have an exact optimal solution [vector $\gamma^*(\boldsymbol{u})$] of the simplified problem (5) for a given control vector $\boldsymbol{u}$, then we can easily calculate an approximate optimum $\tilde{\gamma}(\boldsymbol{u}+\Delta\boldsymbol{u})$ of original problems (3) and/or (4), where the required control vector is perturbed slightly, that is, instead of exact $\boldsymbol{u}$, TMF delivers some $\boldsymbol{u}+\Delta\boldsymbol{u}$, where Euclidean norm of vector $\Delta\boldsymbol{u}$ tends to zero as all $\gamma_n^{\min}$ tend to zero. The value of $\phi(\tilde{\gamma})$ [propellant consumption for vector $\tilde{\gamma}(\boldsymbol{u}+\Delta\boldsymbol{u})$] is greater than optimal [in problems (3) or (4)] value $\hat{\phi}(\boldsymbol{u}+\Delta\boldsymbol{u})$, which remains unknown (!). But, involving *dual* variables[13] corresponding to exact solution of the simplified LP problem (5), we can estimate[9,10] the difference by some explicit value $\Delta\phi$, $\phi(\tilde{\gamma}) - \hat{\phi}(\boldsymbol{u}+\Delta\boldsymbol{u}) < \Delta\phi$, which tends to zero as all $\gamma_n^{\min}$ and k tend to zero.

D. General Scheme of Numerical Algorithm

A new effective computational algorithm to solve the whole class of LP problems relevant to TSP (with matrix dimension of approximately 6 × 30) has been proposed as a combination of a new fast algorithm for the following LP problem without upper bound on variables (written here in a vector form):

$$\sum_{n=1}^{N}\gamma_n \to \min_{\gamma} \qquad \text{s.t.}$$
$$\boldsymbol{A}\gamma = \boldsymbol{u}$$
$$\gamma \geqq 0 \tag{6}$$

and the specially modified parametric simplex method[13] as applied to the following parameterized form of linear problem (5), where s is a scalar parameter, $s \in [0, 1]$:

$$\sum_{n=1}^{N}\gamma_n \to \min_{\gamma}, \text{ s.t.}$$
$$\boldsymbol{A}\gamma = s\cdot\boldsymbol{u}$$
$$0 \le \gamma_n \le \gamma_n^{\max} \qquad n = 1:N \tag{7}$$

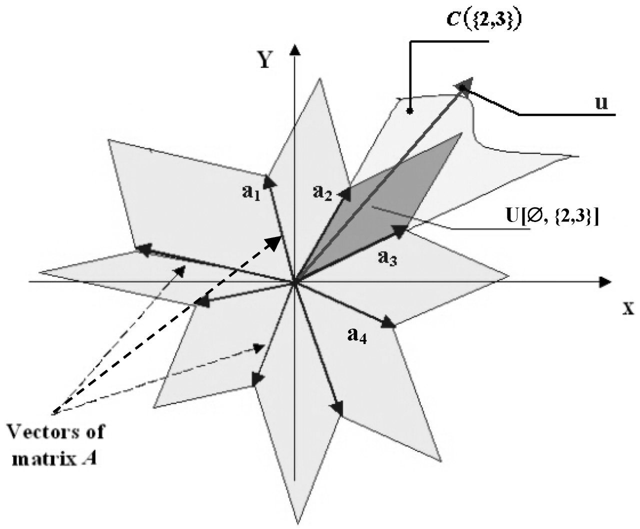


Fig. 1 TMF_Cone algorithm detects the optimal cone in problem (6) (two-dimensional example).

We call this routine TMF_Cone_PS because actually it is a combination of two distinct subroutines: TMF_Cone and TMF_PS. Both subroutines are based on the known[2] "geometrical" representation of the set of all optimal bases (in terms of LP theory) in problems (6) and (5) as, respectively, a finite set of convex polyhedral optimal cones (each cone is formed by M "basic" thrusters) and a finite set of convex optimal parallelotopes (each parallelotope is defined by some subset of "full-on" thrusters and M basic thrusters whose thrust can vary from zero to the upper bound[2]). Both representations can be interpreted as a disjoint (by interiors of optimal cones/parallelotopes) partitioning of sets of all feasible (in appropriate LP problem) control vectors $\boldsymbol{u}$, where each partition member corresponds to some optimal basis of the corresponding LP problem. Two-dimensional ($M = 2$) illustrations of these representations are presented in Figs. 1 and 2.

For example, the cone $C\{2,3\}$ in Fig. 1 represents the set of those two-dimensional control vectors $\boldsymbol{u}$, for which only two thrusters, 2 and 3, should be opened to ensure minimal fuel consumption in problem (6).

As an illustration of a more complex (because of upper bounds on thrusters' activities) problem (5), a parallelotope $U[\emptyset,\{2,3\}]$ in Fig. 1 represents the set of those control vectors for which the same couple of thrusters should be opened (within corresponding $[0, \gamma_n^{\max}]$ intervals). Also in Fig. 2, parallelotope $U[\{2\},\{1,3\}]$ is the set of controls where for any vector $\boldsymbol{u}$ from this parallelotope optimal thrusters combination involves only three thrusters (first, second, and third), and requires the second thruster to be fully opened.

Algorithm efficiency is based most of all on the new fast algorithm (TMF_Cone) to solve unbounded problem (6). The suggested set of algorithms consists of 1) effective new computational algorithm (non-real-time phase) to determine all sets of these cones, including special regularization of thrusters' matrix $\boldsymbol{A}$ to avoid ill-conditioning and degeneracy of problem (6); 2) effective new computational algorithm (non-real-time phase) to construct special tree-like look-up-table (LUT) data-structure; 3) optimized data structures to store this cone complex and LUT in OBC memory; and 4) effective new computational algorithm (real-time phase) to find the optimal cone that contains given control vector $\boldsymbol{u}$ on the basis of LUT data-structure.

TMF_Cone is a result of deep mathematical analysis of optimal basis's structure in LP problem (6). This algorithm includes a number of programmatic features that lead to its computational effectiveness (in terms of performance and required size of OBC memory). For example, at the non-real-time phase the algorithm calculates normal vectors to facets of every optimal cone detected. According to known results of linear algebra, these normal vectors can be chosen as rows of inverse basis matrices, thus the real-time phase of the algorithm does not spend time to calculate these inverse matrices again (that a standard simplex method would do implicitly).

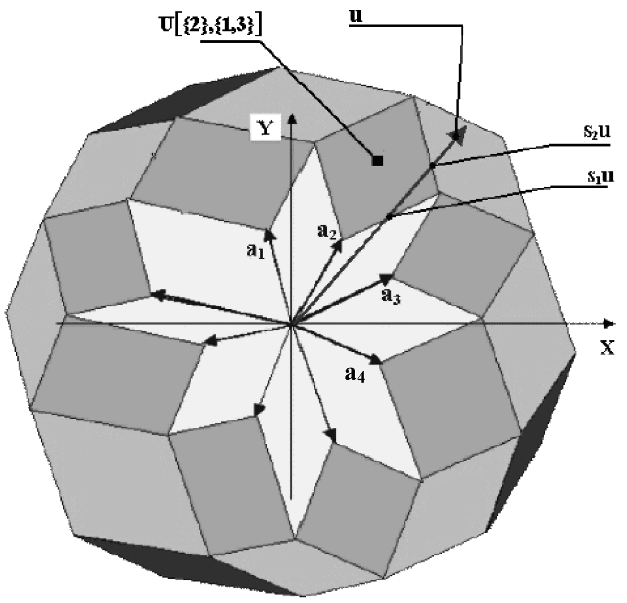


Fig. 2 TMF_PS runs over composition of optimal parallelotopes: controllability envelope of Eq. (5) (two-dimensional example).

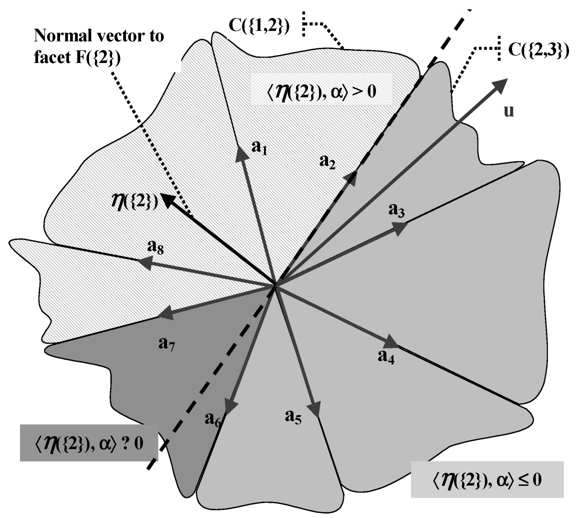


Fig. 3 Two-dimensional illustration of TMF_Cone algorithm idea.

One of the main ideas of TMF_Cone is presented in Fig. 3. Take some normal vector to some facet of some optimal cone, for example, consider cone $C(\{1,2\})$, its facet $F(\{2\})$, and its normal vector $\boldsymbol{\eta}(\{2\})$. In the two-dimensional example there are two one-dimensional facets formed by vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, and facet $F(\{2\})$ is a ray formed by vector $\boldsymbol{a}_2$. The normal vector is perpendicular to $\boldsymbol{a}_2$ and is normalized to have $\langle \boldsymbol{\eta}(\{2\}), \boldsymbol{a}_1 \rangle = 1$ (scalar product of two vectors at the left side of the preceding equality). We can easily see that this normalization means that $\boldsymbol{\eta}(\{2\})$ is a first row of a 2×2 matrix $\{\boldsymbol{a}_1, \boldsymbol{a}_2\}^{-1}$, which is inverse to 2×2 matrix $\{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ whose columns are vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$; the preceding normalization does not help to illustrate the main idea, but it is used by TMF_Cone to increase performance of real-time phase of algorithm.

Vector $\boldsymbol{\eta} = \boldsymbol{\eta}(\{2\})$ and corresponding hyperplane $\{\alpha \in R^M : \langle \boldsymbol{\eta}, \boldsymbol{\alpha} \rangle = 0\}$ (dashed line in the figure) enable the division of the whole set of basic cones into three groups: 1) "positive" cones lying completely in positive (relatively to direction of $\boldsymbol{\eta}$) half-space, $C(\{1,2\})$, $C(\{1,8\}$, $C(\{7,8\})$ in the figure; 2) "negative" cones lying completely in negative half-space, $C(\{1,2\})$, $C(\{3,4\}$, $C(\{5,6\})$; and 3) "indefinite" cones lying partially in both half-spaces, $C(\{6,7\})$ in the figure. Now, take some control vector $\boldsymbol{u}$ (upper right arrow in Fig. 3). Our purpose is to detect the basic cone that contains a given $\boldsymbol{u}$. At the beginning, we must consider all basic cones as "candidates." Try to reduce the set of possible candidates calculating scalar product $\langle \boldsymbol{\eta}(\{2\}), \boldsymbol{u} \rangle$. If the value of result is negative, we can be sure that vector $\boldsymbol{u}$ does not belong to any cone of positive group, and all of these cones can be eliminated from further investigation; if the result is positive, we can neglect all negative cones. Thus, we substantially reduce the set of possible alternatives as a result of one scalar product operation. Note that an indefinite group of cones remains for further investigation in both cases.

Not going into detail, all steps of TMF_Cone algorithm are similar to what was described in the preceding paragraph.

As to the LUT, it is a kind of binary tree where each nonterminal node corresponds to some facet of some optimal cone, and terminal nodes correspond to the optimal cones themselves. Left and right descending branches of the node correspond to the dissection of the set of optimal cones by a hyperplane corresponding to this facet. At the real-time phase, for the given control vector $\boldsymbol{u}$ the algorithm seeks a path (over the LUT tree) to the corresponding terminal node (optimal cone) as a result of a number of few scalar products of vector $\boldsymbol{u}$ and facet normal vectors corresponding to nonterminal nodes of the LUT tree. Depending on the sign of this scalar product, either the left or right descending branch is selected.

In the real-time mode, TMF_Cone finds a combination of thrusters of not more than 6 (or 7 depending on used modification of TMF_Cone; see more explanations in the following), which provides a requested direction of control vector $\boldsymbol{u}$ with minimal specific propellant consumption.

Note that in practice there are certain constraints regarding the maximum amount of thrusters to be fired at the same time. Nevertheless, it has been assumed in the theoretical study that when multiple thrusters are fired simultaneously there is no degradation in any thruster performance. The preceding constraint could be taken into account by adding a simple routine checking whether the algorithm provides any forbidden combination of thrusters and discarding it.

If the length of vector $\boldsymbol{u}$ is rather small, it is enough to detect an optimal solution to the problem (6); see the two-dimensional illustration in Figs. 1 and 2.

If the length of a given vector $\boldsymbol{u}$ is not small, then the combination of thrusters found by TMF_Cone might become insufficient. It is in this case that the TMF_PS routine might be involved to increase the accuracy of the $\boldsymbol{u}$ length recovering.

TMF_PS is a special modification of the parametric simplex method; it starts computation from an optimal basis [of problem (5), not (6)!] found by fast TMF_Cone algorithm. TMF_PS is an iterative procedure. The value of the scalar parameter $\boldsymbol{s}$ runs over a finite set of nondecreasing values $\{s_{\text{iter}}\}$, $s_{\text{iter}+1} \geq s_{\text{iter}}$. It is a pure algebraic routine based on the analysis of necessary and sufficient optimal conditions of LP problem (5). Nevertheless, it can be well illustrated by representation of the controllability envelope for the problem (5) as a composition of optimal parallelotopes (see Fig. 2), where each parallelotope corresponds to some optimal basis of problem (5) for some vector $\boldsymbol{u}$ (and vice versa: each optimal basis corresponds to some parallelotope).

While $s_{\text{iter}} < 1$, at every iteration TMF_PS routine performs the following operations:

1) It either determines the next (adjacent) optimal parallelotope enabling an increase in the parameter s, if the value of s becomes equal to one, for example, in Fig. 2 it is so for $U[\{2,3\}, \{1,4\}]$, then TMF_PS reports that the optimum for problem (5) has been found.

2) Or it reports that a subsequent increase is impossible; it means that the bound of the controllability envelope has been reached, and problem (5) has no feasible solution for a given vector $\boldsymbol{u}$; roughly speaking, the vector $\boldsymbol{u}$ is too long.

TMF_PS requires a fixed and rather small size of OBC memory to store a kind of its simplex table: 6*(N-6)*sizeof(float), and each iteration of the algorithm requires approximately the same number of arithmetic operations.

There are two modifications (so-called zero level and first level) of the TMF_Cone algorithm. These modifications are distinguished next (see Sec. III) by suffixes −0 and −1 appended to all designations corresponding to TMF_Cone (e.g., ATSD-0, ATSD-1, TSD-0, TSD-1).

Without going into detail, the zero-level modification is optimized to handle small-sized control vectors $\boldsymbol{u}$. The first-level modification is optimized to handle middle-sized control vectors $\boldsymbol{u}$ because it can find optimal combination of up to seven thrusters providing required direction of vector $\boldsymbol{u}$. As a result, the first-level TMF_Cone can save a number of iterations of the TMF_PS algorithm, which is slower than TMF_Cone. This feature enables an increase in real-time performance of TMF_Cone_PS (less computation for the middle-size controls and/or extension of controllability envelope), but at the expense of additional OBC memory consumption. In particular, for a propulsion subsystem having 16 thrusters the size of TSD-0 is about 26 KB, and the size of TSD-1 is about 85 KB; in the case of 28 thrusters (e.g., like ATV[11]), corresponding numbers for TSD-0 and TSD-1 are 115 and 390 KB, respectively (with no data compression).

The general scheme of real-time onboard computation on the basis of TMF_Cone_PS algorithm can be described as follows:

1) TMF_Cone detects optimal thruster activity providing requested direction of $\boldsymbol{u}$.

2) If necessary, TMF_PS detects the optimal thruster combination providing more length along the direction of $\boldsymbol{u}$ ($s^* \leq 1$).

3) Discretization of the TMF_Cone_PS output provides correct inputs for thrusters' electronics.

III. Software Implementation

In Fig. 4, a generalized architecture of the software is depicted realizing functionality described in Sec. II. It is called the spacecraft thruster management subsystem design and analysis software tool (STM-DAT).

The following three modes are selectable by the user: 1) mode of computation of all thruster selection data (ATSD) in non-real-time (presimulation mode); 2) mode of preparation of the thruster selection data (TSD-R) in the form suitable for real-time mode; and 3) mode of evaluation and analysis of the simulation results.

Computation of ATSD is performed in non-real-time mode. The results of execution of the module of all raw data computation are stored in the repository of ATSD. Each of the ATSD files can contain the following:

1) The first is level zero of ATSD (ATSD-0), which will restrict itself to finding the optimal combination of not more than six thrusters to be fired simultaneously. This number has been selected to ensure six-degree-of-freedom control of the spacecraft.

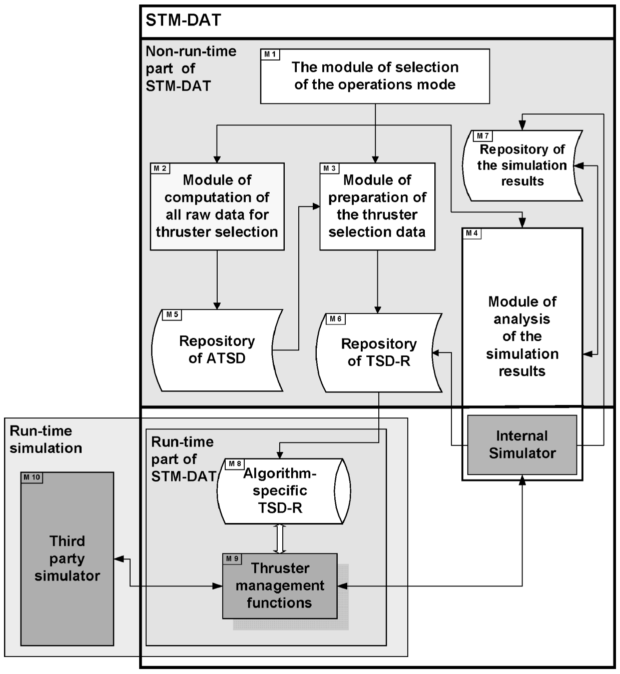

STM-DAT
Non-run-time part of STM-DAT
M 1
The module of selection of the operations mode
M 2
Module of computation of all raw data for thruster selection
M 3
Module of preparation of the thruster selection data
M 7
Repository of the simulation results
M 4
Module of analysis of the simulation results
M 5
Repository of ATSD
M 6
Repository of TSD-R
Run-time simulation
Run-time part of STM-DAT
M 8
Algorithm-specific TSD-R
M 10
Third party simulator
M 9
Thruster management functions
Internal Simulator

Fig. 4 High-level architecture of STM-DAT software.

2) The next is level zero plus level one of ATSD (ATSD-1), which enables fast real-time routines to find the optimal combination of seven thrusters. By adding an additional thruster (as compared with ATSD-0), ATSD-1 allows for expanding of the controllability envelope (see Fig. 3), though at the expense of additional OBC memory (required for a bigger LUT).

Based on ATSD, the algorithms embedded in the module of preparation of TSD-R form a set of data that will be used by TMF in the real-time mode. The preparation of TSD-R is performed in non-real-time mode. These final data form the repository of TSD-R.

During TMF software execution onboard the spacecraft, the main routine implies the fast search of the optimal solution through the LUT tree. This could be done using either ATSD-0 or ATSD-1 databases. From the practical perspective, ATSD-0 or ATSD-1 could be used in nominal situations to control a spacecraft also in some limited number of foreseen failure situations, when these data could be precomputed. In more complicated off-nominal situations, the parametric simplex routine can be invoked after ATSD-0 or ATSD-1 if it is necessary to increase the accuracy of the solution being searched for. As an example, a collision-avoidance maneuver (during the final approach to the space station) could be mentioned when it is of critical importance to provide as "long" as possible thrust vector along some direction that might be not precisely known in advance.

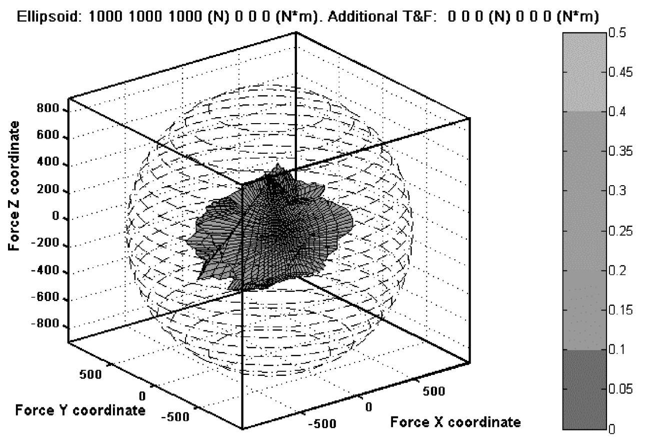
In the discrete mode, real-time software performs rounding of continuous solution to compute the time of functioning of each thruster in such a way that it would be an integer number multiplied by the minimum open thruster interval. It gives acceptable results in the case when $\tau^{\min}$ is much less than sampling period $\boldsymbol{T}$.

IV. Test Results

A. Spacecraft Controllability Envelopes

Simulations have been performed to test performance and functionality of the implemented family of thruster management algorithms. In these simulations, mass-inertia properties of spacecraft and configuration of its thruster subsystem are close to those of the ATV[11] (28 thrusters). Figures 5–8 illustrate test results by visualizing the full six-dimensional controllability envelope through its three-dimensional "generalized" cross sections done by a three-dimensional hyperplane that corresponds to the case when three other components of six-dimensional control vector are fixed. More precisely, what are seen in these figures are inner approximations of real cross sections because of the fixed upper limit on TMF_PS iterations.

Before running each test, the STM-DAT tool prompts the user to split the control vector components (three force and three torque)



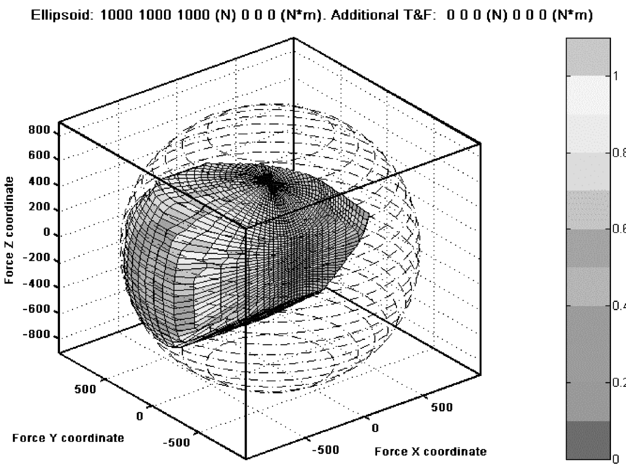
Estimation of TMF parameters:

τ_{mean}=0.069 (msec);

s_{mean}=0.334; s_{min}=0.177;

consumption$_{mean}$=0.274(kg); consumption$_{max}$=0.402(kg).

Fig. 5 Example of force controllability envelope (TMF_Cone-0 only, zero torques).



Estimation of TMF parameters:

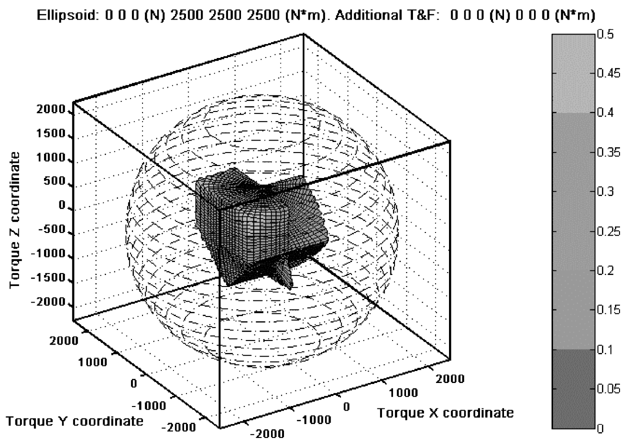
τ_{mean}=0.170(msec); τ^{iter}_{mean}=0.024(msec);

s_{mean}=0.538; s_{min}=0.256;

$iter^{max}_{PS}$=10.00; $iter^{mean}_{PS}$=7.19

consumption$_{mean}$=0.655(kg); consumption$_{max}$=1.021(kg).

Fig. 6 Example of force controllability envelope (TMF_Cone-0+PS, zero torques).



Estimation of TMF parameters:

τ_{mean}=0.078 (msec);

s_{mean}=0.375; s_{min}=0.270;

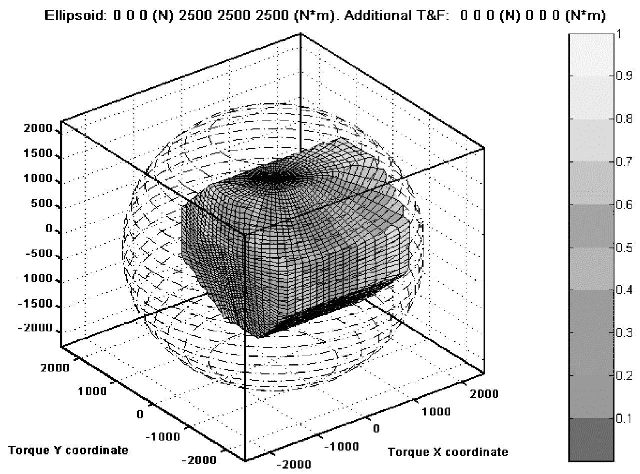
consumption$_{mean}$=0.244(kg); consumption$_{max}$=0.467(kg).

Fig. 7 Example of torque controllability envelope (TMF_Cone-0 only, zero forces).

into two parts, so-called free and fixed (referred to as "additional" in figures). The free components vary over surface of the three-dimensional ellipsoid, which is dashed in figures. The user can set the required length of half-axes of the ellipsoid. The three additional components can be set to any fixed values. For example, the user can choose all three torque components as free ones and fix forces as $F_x = 100$ N, $F_y = F_z = 0$. This would mean that the tool should build up a three-dimensional torque controllability envelope with an additional constraint of always maintaining $F = 100$ N along the spacecraft X axis.

Inside the dashed ellipsoids one can see the output ("solid bodies") provided by the implemented software. The vertical bar on the right is illustrating propellant consumption and allows the user to quickly identify fuel-thirsty directions.

Pairs of figures {Figs. 5 and 6} and {Figs. 7 and 8} illustrate how usage of TMF_PS improves the controllability envelope (in comparison with results of TMF_Cone usage only). Each



Estimation of TMF parameters:

τ_{mean}=0.138(msec); τ^{iter}_{mean}=0.019(msec);

s_{mean}=0.601; s_{min}=0.395;

$iter^{max}_{PS}$=10.00; $iter^{mean}_{PS}$=7.21

$consumption_{mean}$=0.567(kg); $consumption_{max}$=0.931(kg).

Fig. 8 Example of torque controllability envelope (TMF_Cone-0+PS, zero forces).

pair corresponds to the same settings of free and fixed control components:

1) For the first pair, all three translation components are free and vary over the sphere of radius 1000 N; all torques are fixed to zero.

2) For the second pair, all three translation components are fixed to zero, and all torques are free and vary over the sphere of radius 2500 N · m.

It can be easily seen that the volume of inner approximations of the controllability envelope presented in Figs. 5 and 7 (TMF_Cone only) is much less than respective output achieved as a result of combination of TMF_Cone and TMF_PS algorithms (Figs. 6 and 8; maximum number of TMF_PS iterations is equal to 10).

B. Comparative Tests

Comparative simulations were performed to test the capabilities of the designed and implemented algorithm when different combinations of force and torque vectors were used as input data. As an opponent algorithm, a standard "primal" simplex method (SIMPLX[14]) was tested along with its TMF-specific software implementation available at ESA since the initial phase of the ATV program (28 thrusters).

A sine-wave model vs time forms input force and torque vectors. There were always six input waves with different amplitudes and periods, but because of limited space only a single time diagram is represented in Fig. 9. Here, a solid sine wave represents the input process, the dashed line corresponds to results of the standard-simplex-based TMF, and the dash-dot line corresponds to the new TMF described in this paper (namely, TMF_Cone-0+PS), and upper limit of TMF_PS iterations is equal to 10. Force is measured in Newtons (vertical axis), and time is measured in seconds (horizontal line).

The preceding diagram clearly demonstrates advantages of the designed algorithm, in the case when the six input waves produce infeasible requirements (i.e., too long input six-dimensional vector $\boldsymbol{u}$). Let us, for example, consider the instant $t = 3.8$ s. At this moment the value of the required input force is -1500 N (it is beyond maximum capabilities of the thruster subsystem), whereas the simplex-based TMF returns the value of $+250$ N. So, instead of driving the spacecraft in the $-Y$ direction, the force would be applied along the opposite $+Y$ axis! Of course, this irrelevant result is caused by absence of a feasible solution to the corresponding LP problem, and a standard simplex is not applicable in this case.

Conversely, the designed and implemented algorithm (dash-dot line) demonstrates much better behavior. Of course, it does not follow the peaks of the input process either (just because it is impossible), but it always returns the output force/torque vector directionally aligned with the input one. And integrally, the dash-dot line is closer to the reference (solid) curve than the dashed one. Note that the different reaction of both algorithms for the same input shown in Fig. 9 (e.g., at $t = 4$ and 9 s) is because inputs along five other components (not shown in the plot because of limited space) were quite different (all six components are not phase synchronized). Accordingly, the resources of the propulsion subsystem were allocated in a different way near periodically repeated values of input wave shown as solid sinusoid.

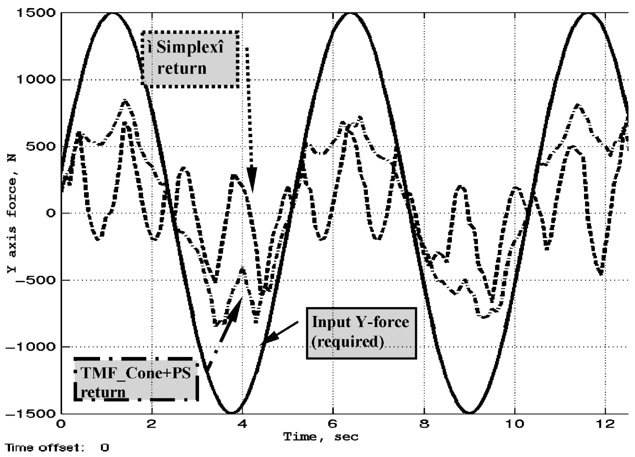


Fig. 9 Comparative simulations (presented algorithm vs standard simplex).

Implemented TMF software was also tested in a closed-loop mode together with a simulation of spacecraft orbital motion. Different kinds of maneuvers were tested such as attitude stabilization, angular reorientation, and translation, including rendezvous and docking with the ISS that required six-dimensional control at its final stage. In all cases the algorithms implemented in the reported study demonstrated stability and robustness.

During algorithm and software design phase of the project, a number of comparative tests have been performed for dimensions $M = 6$ and $N = 16, 20, 28$ and thruster matrices typical for ATV spacecraft. Competitive implementations included MATLAB® LINPROG routine run with following options: 1) large-scale option that switches on interior-point method (LIPSOL[7]); 2) medium-scale option involving reduction of LP problem to quadratic programming and usage of a kind of quasi-Newton method to solve this problem; and 3) simplex option involving standard simplex-method (available in MATLAB 7).

It was found that both LIPSOL and simplex routines demonstrated 1) the same "unpleasant" behavior in the case of infeasible constraints (when a too long control vector was required), and 2) a worse performance. They were up to 10 times slower than TMF_Cone_PS routine for "small" control vectors (when TMF_Cone and, maybe, one or two TMF_PS iterations were enough to find the optimal solution) and three to four times slower for "long" control vectors (when TMF_PS performed about 10 iterations). In the case of the medium-scale option, there were even worse results in cases of infeasible constraints (it returned substantially negative values of variables subjected to nonnegative constraints) and also worse performance in all cases (it was better than for large-scale LIPSOL, but at least two to three times slower than TMF_Cone_PS).

C. Onboard Implementation Issues

Some integral results of real-time simulation are represented in Table 1 for force envelope (module of required force is 1000 N along each of three components). Computation time values (for real-time mode) are presented for different options of the TMF algorithm (prefix TMF_ is omitted). Represented tests were conducted on a Sun-Blade (550-MHz) workstation. Roughly, assuming that existing onboard computers are 20–30 times slower than the preceding workstation, we get an estimate of 4–6 ms for real-time slot for the hardest case when 10 iterations of the PS algorithm are needed. Data

Table 1 TMF_Cone algorithm performance[a]

Performance characteristic	Cone-0	Cone-1	Cone-0+PS	Cone-1+PS
Average time of algorithm execution, ms	0.071	0.094	0.200	0.204
Average time of one step of PS algorithm, ms	N/A	N/A	0.024	0.03
Average number of iterations of PS algorithm	N/A	N/A	5.46	4.29

[a]N/A = not applicable.

in Table 1 correspond to the configuration of a spacecraft with 20 thrusters. However, values for the configuration with 28 thrusters do not increase dramatically.

Regarding the size of memory to store the data to be used in real-time computations onboard the spacecraft, for a propulsion subsystem having 16 thrusters the size of ATSD-0 set is about 26 KB, and the size of ATSD-1 is about 85 KB. In the case of 20 thrusters, corresponding sizes of ATSD-0 and ATSD-1 are ~52 and ~194 KB, respectively. In the case of 28 thrusters (like in ATV spacecraft), we have ~115 KB for ATSD-0 and ~390 KB for ATSD-1. These variations are nonlinear; specific functions were not studied, but we can see that the required size of memory remains acceptable for the state-of-the-art OBC.

It is obvious that even a simple compression technique would reduce the preceding numbers significantly, but there was no attempt at this stage of the study to compress these data.

The concept based on a precomputed set of optimal solutions suggested by the authors of this paper was taken into account by the designers of the ATV control system (EADS Space Transportation) who implemented their own algorithms and flight software.[15] The concept implemented for the ATV is, to some extent, similar to the TMF_Cone-0 described in this paper. However, the original mathematical task formulation in the paper[15] was simplified as compared with the present paper, and nothing similar to the LUT approach was implemented, although usage of LUT would help considerably to reduce computational time required by the OBC. Also, the present approach is much broader than the one described in Ref. 15 because it suggests several levels of ATSD and a special modification of parametric simplex, which starts computations based on the solution already selected by TMF_Cone.

V. Conclusions

Original mathematical algorithms related to the thruster selection problem have been designed, implemented, and tested covering both non-real-time and real-time computations. The suggested concept is applicable to a broad range of propulsion systems, including continuous and discrete cases to control open thruster intervals. Software implemented on the basis of designed algorithms demonstrated effectiveness and could be used in state-of-the-art onboard computers.

Acknowledgments

The research reported in this paper was performed under ESA/ESTEC Contract 15789/02/NL/LvH. The authors of this paper express their gratitude to all experts involved in this study.

References

[1]Bergmann, E. V., Croopnick, S. R., Turkovich, J. J., and Work, C. C., "An Advanced Spacecraft Autopilot Concept," *Journal of Guidance and Control*, Vol. 2, No. 3, 1979, p. 161.

[2]Crawford, B. S., "Configuration Design and Efficient Operation of Redundant Multi-Jet Systems," *Proceedings of AIAA Guidance, Control, and Flight Mechanics Conference*, AIAA, New York, 1969; also AIAA Paper 69-845, Aug. 1969.

[3]Paradiso, J. A., "A Highly Adaptable Steering/Selection Procedure for Combined CMG/RCS Spacecraft Control, Detailed Report," C. S. Draper Lab., Report CSDL-R-1835, Cambridge, MA, March 1986.

[4]Paradiso, J. A., "Application of Linear Programming to Coordinated Management of Jets and Aerosurfaces for Aerospace Vehicle Control," C. S. Draper Lab., Report CSDL-R-2065, Cambridge, MA, Nov. 1988.

[5]Paradiso, J.A., "A Highly Adaptable Method of Managing Jets and Aerosurfaces for Control of Aerospace Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 1, 1991, pp. 44–50.

[6]Mehrotra, S., "On the Implementation of a Primal-Dual Interior Point Method," *SIAM Journal on Optimization*, Vol. 2, No. 4, 1992, pp. 575–601.

[7]Zhang, Y., "Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment," Dept. of Mathematics and Statistics, Univ. of Maryland, TR96-01, Baltimore, MD, July 1995.

[8]Wright, S. J., "Applying New Optimization Algorithms to Model Predictive Control," *Chemical Process Control-V, CACHE*, AIChE Symposium Series No. 316, edited by J. Kantor, C. Garcia, and B. Carnahan, Vol. 93, American Inst. of Chemical Engineers, New York, 1997, pp. 147–155.

[9]Levitin, E. S., *Perturbation Theory in Mathematical Programming and Its Applications*, Wiley, New York, 1994, pp. 261–322.

[10]Levitin, E. S., and Voloshinov, V. V., "Approximate Search for a Global Minimum in Problems of Mathematical Programming That Are Close to Convex," *Computational Mathematics and Mathematical Physics*, Vol. 39, No. 3, 1999, p. 365.

[11]Gonnaud, J. L., and Pascal, V., "ATV Guidance, Navigation and Control for Rendezvous with ISS," Spacecraft Guidance, Navigation, and Control Systems, *Proceedings of the 4th ESA International Conference*, edited by B. Schurmann, SP-425, ESTEC, Noordwijk, The Netherlands, 2000, p. 501.

[12]Fehse, W., Vankov, A., Shlyaev, P., Ankersen, F., and Alyoshin, A., "Remote Intervention in Automatic Onboard GNC Systems," *Proceedings of 3rd International Conference on Spacecraft GNC Systems*, SP-381, ESTEC, Noordwijk, The Netherlands, 1996, pp. 109–118.

[13]Murtagh, B. A., *Advanced Linear Programming: Computation and Practice*, McGraw–Hill, New York, 1981.

[14]Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (eds.), *Numerical Recipes in C, The Art of Scientific Computing*, 2nd ed., Cambridge Univ. Press, Cambridge, England, U.K., 1992, pp. 430–443.

[15]Martel, F., "Optimal Simultaneous 6 Axis Command of a Space Vehicle with a Precomputed Thruster Selection Catalogue Table," *Advances in the Astronautical Sciences*, Vol. 116, Aug. 2003, pp. 1363–1374.