Interior-Point Algorithms for Control Allocation

John A. M. Petersen*
Raytheon Missile Systems
Tucson, AZ 85706

Marc Bodson†
University of Utah
Salt Lake City, UT 84112

Abstract-- **The paper considers linear programming formulations of control allocation problems, including those associated with direct allocation and mixed ℓ_1-norm objectives. Primal-dual and predictor-corrector path-following interior-point algorithms, that are shown to be well-suited for the control allocation problems, are described in some detail with an emphasis on preferred implementations. The performance of each algorithm is evaluated for computational efficiency and for accuracy using linear models of a C-17 transport and a tailless fighter aircraft. Appropriate choices of stopping tolerances and other algorithm parameters are studied. Comparisons of speed and accuracy are made to the simplex method. Results show that real-time implementation of the algorithms is feasible, because they do not require an excessive number of computations.**

Index Terms—**Control allocation, flight control, linear programming, interior-point methods.**

* Principal Systems Eng., GNC Center, Raytheon Missile Systems, 1151 E. Hermans Rd, Tucson, AZ, 85706, Member AIAA.
† Professor, Electrical & Computer Engineering, University of Utah, 50 S Central Campus Dr Rm 3280, Salt Lake City, UT 84112, U.S.A, Senior Member AIAA.

INTRODUCTION

Flight control laws are usually designed without accounting for constraints on the control effectors. *Control allocation* algorithms can be used to distribute the control law requirements among multiple effectors in some optimal manner while accounting for their limited range. The objective is typically to minimize the error between the commanded acceleration, generated by the control law, and an achievable acceleration. Other criteria for minimizing drag, power, actuator deflection, etc., may also be included. Two objectives considered in this paper are direct allocation[1,2,3] and mixed ℓ_1-norm optimization.[4] Direct allocation solves for a vector of control variables that produce an acceleration vector with a direction identical to the desired acceleration vector, while minimizing the magnitude of their difference. Mixed optimization minimizes the acceleration error together with other secondary terms, such as control effort.

Enns[5] and Buffington[6] have successfully transformed the control allocation problem from an ℓ_1-norm error minimization objective into standard linear programming format and have implemented solutions using commercially available software. Linear programming algorithms may be categorized in two groups, namely, simplex algorithms and interior-point algorithms. Simplex algorithms are well-known and have been previously tested for control allocation.[4,5] Iterates of the simplex method travel along the edges of the feasible space while decreasing the value of the cost function at each iteration. One drawback is the simplex method's susceptibility to cycling due to

degeneracy or roundoff error. Anti-cycling rules may help but come with no guarantees. Interior-point methods differ from simplex algorithms because they always progress towards the optimal solution from within the feasible space, rather than along the boundary. Interior-point algorithms have good convergence properties, which may make them better suited to the fixed-time implementations of safety-critical systems. Although not generally observed in practice, simplex methods cannot guarantee that a feasible solution is anywhere near the optimum if the algorithm is stopped prematurely. Interior-point methods, on the other hand, can be viewed as a gradient method that always progresses in the general direction of the optimal point.

This paper starts from linear programming formulations of the control allocation problem. Because interior-point algorithms are numerous and the literature is somewhat discouraging to the non-specialist, details of two standard approaches suited for the control allocation problem are described with an emphasis on implementation. These algorithms, called primal-dual path-following, and predictor-corrector path-following algorithms, are adapted to the control allocation problems and tested on models of a C-17 transport aircraft and an advanced tailless fighter. Simplex solutions are computed for a baseline comparison. Guidelines for obtaining weighting parameters and stopping tolerances are derived from the results of the tests. Primal-dual path-following is shown to be the most efficient of the two interior-point algorithms for the control allocation problems presented.

PROBLEM STATEMENT

The objective of the control allocation problem is to determine the n components of the control vector, u, that result in the desired roll, pitch, and yaw components of the acceleration vector, $a_d \in \Re^m$. We assume that they are related by the control effectiveness matrix CB as follows:

$$a_d = CBu \tag{1}$$

where $CB \in \Re^{m \times n}$. We also assume that the vector is constrained by upper and lower bounds so that

$$u_{\min} \le u \le u_{\max} \tag{2}$$

The vector u may be composed of the most restrictive components of the rate limits of a single control cycle and the position limits. Model reference control laws[7] and dynamic inversion control laws[8] allow one to specify the trajectories of the output of the system by selecting the value of the term CBu due to the control input. For underdetermined systems, where $m < n$, control allocation algorithms are necessary to find the control vector u that achieves that goal. However, note that there is no guarantee that an arbitrary a_d is attainable or that the solution is unique. Although there is no inherent restriction on the values of m and n for interior-point algorithms, the case where $m < n$ corresponds to the reality of the problems under consideration.

Direct Allocation Formulation

The objective optimized by direct allocation can be expressed as:

$$\begin{aligned} &\max_{\zeta} \quad \zeta \\ &\text{subject to} \quad CBu = \zeta a_d, \; u_{\min} \le u \le u_{\max} \end{aligned} \tag{3}$$

The solution to (3) is a control vector that results in an acceleration proportional to a_d and maximally achievable. Existence of the solution is guaranteed if $u = 0$ belongs to the constraint set.

Mixed ℓ_1-norm Formulation

Error norm formulations enable the minimization of the control error without consideration for the direction of the acceleration vector. In general, there is no guarantee that a_d is attainable or that the solution is unique. If the solution is not unique, a secondary objective is to minimize the magnitude of the control vector or its distance from a preferred control value, u_0. Combining the two objectives is known as mixed optimization. A mixed objective offers additional flexibility for a designer to satisfy secondary objectives. It is often desirable to have the control vector approach a preferred control value that reduces drag, for instance. The mixed ℓ_1-norm objective is defined to be

$$\begin{aligned} &\min_{u} \quad \left\| a_d - CBu \right\|_1 + h \left\| u - u_0 \right\|_1 \\ &\text{subject to} \quad u_{\min} \le u \le u_{\max} \end{aligned} \tag{4}$$

where *h* is a scalar weighting factor and u_0 is a preferred control. The ℓ_1-norm is used to facilitate conversion to linear programming format.

Direct Allocation vs. Error Minimization

Preserving the direction of the desired acceleration does not, in general, minimize the acceleration error for unattainable commands. The direct allocation solution will lie on the boundary exactly in line with the desired vector. However, the solution with minimum error results in the smallest absolute difference between an attainable acceleration and the desired one. Which solution is preferable is debatable, although some piloted simulations have demonstrated a more stable "feel" with error minimization.[9]

Standard Form of Linear Programs

To make use of linear programming methods to solve the control allocation problems (3) and (4), they can be expressed in the form of

$$\begin{aligned} \min_{x} \quad & J_P = c^T x \\ \text{subject to} \quad & Ax = b,\ 0 \le x \le x_{\max} \end{aligned} \tag{5}$$

which is a standard primal linear programming problem description.

Feasibility and Infeasibility

Consider the general optimization problem

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & G(x) = 0,\ H(x) \ge 0 \end{aligned} \tag{6}$$

with smooth functions $f : \Re^n \to \Re$, $G : \Re^n \to \Re^m$, and $H : \Re^n \to \Re^n$. The values of $x \in \Re^n$ that satisfy the constraints are *feasible*. If a feasible solution exists, then at least one optimal solution also exists. Otherwise, the problem is called *infeasible*. The term

infeasible is also used to refer to values of x that do not satisfy the equality constraints. The term *interior* refers to values of x that are strictly within the inequality constraints, which does not include boundary points. An infeasible interior-point algorithm is one that accepts starting points that comply with the inequality constraints but not necessarily with the equality constraints. The control allocation problems as stated in (3) and (4) are guaranteed to have feasible solutions. The mixed ℓ_1-norm is clearly feasible by selecting any control u within the bounds $u_{\min}$ and $u_{\max}$. For direct allocation, the problem is feasible if $u_{\min} \le 0 \le u_{\max}$.

LINEAR PROGRAMMING FORMULATIONS OF CONTROL ALLOCATION

In order to take advantage of linear programming algorithms, the problems (3) and (4) must be recast in a linear programming format, such as (5). The results of the conversion processes developed by Bodson[4] are given below for convenience.

Direct Allocation

The direct allocation problem is already in standard form, however, an equivalent problem of reduced dimension can be solved, creating a significant computational savings (about 1/3rd reduction per iteration for the models in this paper). Direct allocation can be expressed in the form of (5) by making the following assignments:

$$\begin{aligned} A &= M \cdot CB, & b &= -Au_{\min} \\ c^T &= -a_d^T CB, & x_{\max} &= u_{\max} - u_{\min} \end{aligned} \tag{7}$$

where the new variable is defined as $x = u - u_{\min}$. For $m = 3$, the matrix, M, is defined as $M = \begin{bmatrix} a_{d,2} & -a_{d,1} & 0 \\ a_{d,3} & 0 & -a_{d,1} \end{bmatrix}$ with the rows of CB and a_d reordered so that the first

element of $a_d = \begin{bmatrix} a_{d,1} & a_{d,2} & a_{d,3} \end{bmatrix}^T$ is the one with largest magnitude. For the trivial case where $a_d = 0$, the solution is u =0, otherwise, the control vector is obtained from

$$u = x + u_{\min} \tag{8}$$

If $\zeta = \dfrac{a_d^T CBu}{a_d^T a_d} > 1$, the control is scaled to an achievable value $u = u/\zeta$.

Mixed ℓ_1-norm Objective

The objective function for the mixed ℓ_1-norm problem can be transformed into the following linear programming formulation

$$\begin{aligned} \min_x \quad & \begin{bmatrix} h \cdots h & h \cdots h & 1 \cdots 1 & 1 \cdots 1 \end{bmatrix} x \\ \text{subject to} \quad & \begin{bmatrix} CB & -CB & I & -I \end{bmatrix} x = \begin{bmatrix} a_d - CBu_0 \end{bmatrix} \\ & 0 \le p_+ \le u_{\max} - u_0, \qquad 0 \le p_- \le u_0 - u_{\min} \\ & v_+ \ge 0, \; v_- \ge 0 \end{aligned} \tag{9}$$

The variables are combined in the vector, $x = \begin{bmatrix} p_+^T & p_-^T & v_+^T & v_-^T \end{bmatrix}^T$. Once the optimal x is determined, the control can be recovered with

$$u = p_+ - p_- + u_0 \tag{10}$$

It should be noted that the problem may be reduced if any element(s) $u_{0,i} = u_{\max,i}$ or $u_{0,i} = u_{\min,i}$ for i=1…n. In this case, the corresponding column(s) of CB or $-CB$ in (9) are removed.

INTERIOR-POINT ALGORITHMS

Linear programming techniques are desirable because of the existence of well-established, reliable algorithms. They can solve any problem that can be transformed

into a linear cost function with linear constraints, but the size of the typical control allocation problem is rather small. Much of the overhead found in commercial linear programming code is included to handle a wide variety of linear programming problems, and many of the added features are not beneficial to the control allocation problem. The emphasis of these programs is on very large-scale, sparse problems with thousands of variables, whereas current control allocation problems are small-scale, dense problems. Using commercial solvers makes comparison of different techniques difficult because the user is generally at the mercy of the linear programmer's choice in optimization code. Algorithms are usually tested against a group of problems known as the NETLIB suite. These are problems of many different sizes and difficulty. No algorithm has yet been shown to be universally faster for every linear programming problem in the NETLIB suite, and the only way of knowing which one is faster for a given problem is to test it specifically for that problem.

The main point of this section is to familiarize the reader with interior-point algorithms and their implementation for control allocation. This section focuses on one particular algorithm know as primal-dual path-following. A variation called predictor-corrector path-following is also included. There are many adaptations of each method, but our discussion is limited to standard implementations with minor refinements suited to our problems. The concept of a path that interior-point solutions follow was introduced by Megiddo.[10] A natural progression from primal-dual or 1st order path-following algorithms is to extend it to 2nd order. This was done by Mehrotra[11] and developed (popularized) further by Lustig, Shanno, and Marsten.[12] The method is often

referred to as a predictor-corrector. The derivations of these algorithms are documented in more detail in other papers, and the presentation is limited here to what is necessary for implementation.

Primal-Dual Path-Following

The primal-dual path-following algorithm is described below and is captured in pseudo-code in Figure 1 at the end of the section. The algorithm can be derived from either the *primal* or *dual* linear programs. Let the primal problem, with objective J_P, be expressed as

$$\begin{aligned} &\min_x \; J_P = c^T x \\ &\text{subject to } \; Ax = b, \quad x \geq 0 \end{aligned} \tag{11}$$

Incorporation of upper bounds on *x* are discussed later. The search for a lower bound on J_P leads to the development of the dual problem

$$\begin{aligned} &\max_\lambda \; J_D = b^T \lambda \\ &\text{subject to } \; A^T \lambda + s = c, \; s \geq 0 \end{aligned} \tag{12}$$

The difference between primal and dual cost functions plays an important role in the development of the primal-dual algorithms. The duality principle, $c^T x \geq b^T \lambda$, credited to John von Neumann,[13] may be applied generally to primal and dual objectives so that $J_P \geq J_D$ with equality only when the primal and dual variables are optimal. Since the cost function for the dual problem provides a lower bound on J_p, the difference, called the duality gap, between the values of the primal and dual objectives can be used to determine how close a solution is to the optimal point.

Optimality

The Karush-Kuhn-Tucker[14,15] (KKT) optimality conditions specify that optimality requires primal and dual feasibility, and complementarity. That is, the pair (x,s) is a globally optimal solution if and only if

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ Xs &= 0 \\ x \geq 0, s &\geq 0 \end{aligned} \tag{13}$$

where X=diag(x). The first set of equations in (13) enforces dual feasibility, the second enforces primal feasibility, and the third enforces the complementarity conditions.

A similar set of equations can be derived from the unconstrained equivalents to (11) and (12). In the primal framework, the non-negativity constraints on x are approximated with a smooth barrier function, such as $log(x)$, which approaches -∞ as x approaches zero. The equality constraints can be included in the cost function by use of Lagrange multipliers. The resulting unconstrained primal problem is

$$\min_x \quad L = c^T x + \lambda^T \left(b - Ax\right) - \mu \sum_{i=1}^{n} \log\left(x_i\right) \tag{14}$$

with $\mu > 0$. The Lagrange multiplier, λ, is also referred to as a dual variable. In fact, the reverse is also true, that is, the primal variables are the Lagrange multipliers of the dual problem. A system of equations for optimizing (14) can be determined by differentiating L with respect to each variable and setting the results to zero. Making use of the vector $e = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$ and the matrix $X = diag(x)$, the relationships

$$\begin{aligned} c - A^T\lambda - \mu X^{-1}e &= 0 \\ Ax - b &= 0 \\ x &> 0 \end{aligned} \tag{15}$$

are obtained. The variable x is not allowed to be identically zero so that (15) remains well defined. By making the simple substitution, $s = \mu X^{-1}e$, the system of equations in (15) becomes very close to the desired KKT form

$$\begin{aligned} A^T\lambda + s - c &= 0 \\ Ax - b &= 0 \\ Xs - \mu e &= 0 \\ x > 0,\ s &> 0 \end{aligned} \tag{16}$$

If μ is very small, a solution that satisfies (16) will almost be optimal. The constraint $s > 0$ is implied by the substitution just made. It is interesting to note that (16) can be derived from either the primal or dual problems.

The Central Path

Megiddo[10] developed the idea of a central path which lies between the primal and dual solutions. Primal-dual path-following algorithms travel along the proximity of a central path between primal and dual optimal solutions by gradually reducing μ. As μ goes to zero, the solution tends to an optimal point. When on the central path, all products, $x_i s_i$, have the same value, μ, and for this reason it is also known as the complementarity gap. In general, the current point will not be on the central path exactly. In an effort to represent a point closer to the central path, the elements of Xs are reduced to zero at a similar rate. A commonly used estimate of μ is a fraction of the average of these terms and is computed as

$$\mu = \sigma\gamma, \text{ with } \gamma = \frac{x^T s}{n} \text{ and } 0 \le \sigma \le 1 \tag{17}$$

σ is a centering parameter that helps to balance the updates between a pure affine-scaling step $(\sigma = 0)$ and a pure centering step $(\sigma = 1)$. Zhang[16] made σ conditional on the current status of γ. Early on, a large fraction of γ is used to compute μ. Near the end of convergence (when $\gamma << 1$), μ is assigned a smaller and smaller fraction of γ. A dynamic centering parameter that accomplishes this is

$$\sigma = \min(0.1, k\gamma) \tag{18}$$

Experience has shown that $k = 10$ gives the best results for simulations in this paper. Although the duality gap may initially be large, μ decreases exponentially. When it is very small, it decreases at an even faster rate helping convergence to occur very quickly once a solution is near the optimal point.

Step Direction

Since the intent of path-following methods is to iteratively search for the optimal solution along a central path of decreasing values of μ, it follows that a new point $(\lambda + \Delta\lambda,\ x + \Delta x,\ s + \Delta s)$ should also lie in the neighborhood of this path. Following the central path stabilizes the primal-dual algorithm by coaxing each pairwise product $x_i s_i$ to converge to zero at the same rate. For a new point to be on the central path, the equations in (16) become

$$\begin{aligned} A^T(\lambda + \Delta\lambda) + (s + \Delta s) &= c \\ A(x + \Delta x) &= b \\ (X + \Delta X)(s + \Delta s) &= \mu e \end{aligned} \tag{19}$$

The 2nd order terms are considered negligible so that (19) can be simplified to the linearized system of equations, with S=diag(s), to

$$\begin{bmatrix} A^T & 0 & I \\ 0 & A & 0 \\ 0 & S & X \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} c - s - A^T\lambda \\ b - Ax \\ \mu e - Xs \end{bmatrix} \tag{20}$$

Solving (20) gives us the new step direction. The algorithm requires an initial starting point that is strictly in the interior, *i.e.* $(x > 0,\ s > 0)$, but does not require it to be feasible.[17] Residuals may be defined as

$$\begin{aligned} r_c &= A^T\lambda + s - c \\ r_b &= Ax - b \\ r_{xs} &= Xs - \mu e \end{aligned} \tag{21}$$

and the step directions can be simplified to

$$\begin{aligned} \Delta\lambda &= \left(ADA^T\right)^{-1}\left(-r_b + ADr_r\right) \\ \Delta x &= D\left(A^T\Delta\lambda - r_r\right) \\ \Delta s &= -A^T\Delta\lambda - r_c \end{aligned} \tag{22}$$

where $D = \left(X^{-1}S\right)^{-1}$ and $r_r = -r_c + X^{-1}r_{xs}$. D can be defined as $S^{-1}X$, but to be consistent with later expressions of D it is left in this form.

Step Size

To guarantee the next iterate remains in the interior, only a portion of the correction is applied in the update. The primal and dual variables can have independent step sizes so the updates are

$$\begin{aligned} x &= x + \rho\alpha_P \Delta x \\ s &= s + \rho\alpha_D \Delta s \\ \lambda &= \lambda + \rho\alpha_D \Delta\lambda \end{aligned} \tag{23}$$

where $\alpha_P = \min\left\{\left[-\frac{x_i}{\Delta x_i}, 1\right] \middle| \Delta x_i < 0,\ i = 1...n\right\}$, $\alpha_D = \min\left\{\left[-\frac{s_i}{\Delta s_i}, 1\right] \middle| \Delta s_i < 0,\ i = 1...n\right\}$ and $0 < \rho < 1$. Note that ρ is usually chosen above 0.9, and that we use $\rho = 0.99995$ in our implementation.

Stopping Criteria

An optimal solution is a feasible solution that satisfies $J_P = J_D$. Therefore, when the difference between primal and dual cost functions is sufficiently small the algorithm can stop. However, if an infeasible starting point is used, then the stopping rule should include a measure of feasibility also. There is no consensus stopping rule, but all include the duality or complimentarity gaps and primal and dual feasibility measures. We found the following rule sufficient for our tests

$$\max\left(\frac{|J_P - J_D|}{\max\left(|J_P|, |J_D|, 1\right)}, \frac{\|r_b\|_1}{\|b\|_1}, \frac{\|r_c\|_1}{\|c\|_1}\right) \le \varepsilon_s \tag{24}$$

for ε_s around 10^{-3} to 10^{-8} depending on the desired accuracy.

Two-Sided Bounds

Without loss of generality, the lower bound can be assumed to be zero. With an upper bound, a second Lagrange multiplier, *z*, is needed. In addition, the finite upper bounds are handled by use of a separate equality with the help of the non-negative slack variable, *w*. These changes lead to additional equations to (21), namely,

$$\begin{aligned} r_u &= x - w - x_{\max} \\ r_{wz} &= Wz - \mu e \end{aligned} \tag{25}$$

For an infeasible starting point, the step directions are identical to (22) with redefinitions

$$D = \left(X^{-1}S + W^{-1}Z \right)^{-1}$$
$$r_r = -r_c + X^{-1}r_{xs} - W^{-1}r_{wz} + W^{-1}Zr_u \tag{26}$$

and with the additional updates

$$\Delta w = -\Delta x$$
$$\Delta z = -W^{-1}Z\Delta w - W^{-1}r_{wz} \tag{27}$$

with W=diag(w) and Z=diag(z). Updating μ requires γ to be redefined as $\gamma = \dfrac{x^T s + w^T z}{2n}$. Finally, the cost function of the dual problem changes to $J_D = b^T \lambda - z^T x_{\max}$. These differences are of minor computational significance, making the upper bound constraint on x a simple matter.

Starting Point

Careful selection of the starting point can eliminate the need for some of the residuals, but can sometimes lead to slow convergence. Indeed, all interior-point algorithms are sensitive to their starting point and there is no known method that guarantees a good starting point. We have modified slightly a version of a method proposed by Lustig, Marsten, and Shanno.[12] It performs well for all simulations performed for this paper. The idea of the method is to compute a least-squares solution to the equality constraint $Ax = b$ and modify it so that it guarantees a strictly interior-point which is not "too infeasible" and not "too close to a boundary." The cost is slightly less than one iteration. An initial estimate of x is computed as

$$\hat{x} = \left| A^T \left(AA^T \right)^{-1} b \right| \tag{28}$$

The absolute value is used to keep values positive. The norms of b and c are compared with unity to obtain the constants

$$\begin{aligned} n_b &= \max\left(1, \|b\|_2\right) \\ n_c &= \max\left(1, \|c\|_2\right) \end{aligned} \tag{29}$$

The constant p is computed as

$$p = \max\left(0.01 n_b, 100\right) \tag{30}$$

so that it is a minimum value for the primal variables x and w

$$\begin{aligned} x_i &= \max\left(p, \hat{x}_i\right) \\ w_i &= \max\left(p, x_{\max,i} - \hat{x}_i\right) \end{aligned} \tag{31}$$

The free variable λ is set to zero, while each element of s and z are initialized based on the relative size of the corresponding scaling factor in c. Table 1 shows the assignments of s and z which make use of the constant $d = 0.5 n_c$.

Table 1. Initial assignments of the dual variables s and z.

Test	s_i	z_i
$c_i \le -d$	$-c_i$	$-2c_i$
$-d < c_i \le 0$	d	$d - c_i$
$c_i > 0$	$c_i + d$	d

Predictor-Corrector Path-Following

The vast majority of commercial software is based on Mehrotra's predictor-corrector method.[11] Lustig, et al[12] and others further improved upon the method by extending it to

include variables with two-sided bounds. Mehrotra exploited the 2nd order terms of the primal-dual update to provide a better step direction. The update is split into an affine or predictor direction and a centering or corrector direction. Assuming initial infeasibility and including the 2nd order term, the equations in (22) and (27) now become

$$\begin{aligned} A^T\Delta\lambda + \Delta s - \Delta z &= -r_c \\ A\Delta x &= -r_b \\ \Delta x + \Delta w &= -r_u \\ S\Delta x + X\Delta s &= \mu e - Xs - \Delta X\Delta s \\ W\Delta z + Z\Delta w &= \mu e - Wz - \Delta W\Delta z \end{aligned} \tag{32}$$

Consider the step directions for each variable to be composed of two steps, as in

$$\Delta x = \rho\alpha\left(\Delta x^p + \Delta x^c\right) \tag{33}$$

The superscripts, p and c, represent the predictor and corrector portions of the step, respectively, while ρ and α are determined as in (23). The right side of (32) is also partitioned into predictor and corrector portions. The predictor step advances the iterate toward the optimal solution while reducing infeasibility. The residuals, r_b, r_c, r_u, and with r_{xs} redefined as $r_{xs} = Xs$, comprise the predictor portion of the right hand side of (32). The corrector step keeps the updated point near the central path and utilizes the 2nd order terms to make a correction toward the optimal point.

Predictor Step

The predictor step direction is computed with

$$
\begin{aligned}
\Delta\lambda^p &= \left(ADA^T\right)^{-1}\left(-r_b + ADr_r\right) \\
\Delta x^p &= D\left(A^T\Delta\lambda^p - r_r\right) \\
\Delta w^p &= -\left(\Delta x^p + r_u\right) \\
\Delta s^p &= -A^T\Delta\lambda^p - r_c \\
\Delta z^p &= -W^{-1}\left(r_{wz} + Z\Delta w^p\right)
\end{aligned}
\tag{34}
$$

where $r_r = -r_c + X^{-1}r_{xs}$. If a large predictor step can be taken $\left(\min\left(\alpha_D, \alpha_P\right) < 1\right)$ then Δx^c is not computed and is left out of the update. The predictor step sizes are chosen using the same rule for (23).

Corrector Step

The corrector step requires computation of μ. In order to obtain the best estimate of the complementarity gap, it is computed after executing (34). There are many proposed methods of computing μ, but for the control allocation problems, we found that a constant value for the centering parameter, σ, gives the lowest iteration count and also minimizes computations:

$$
\begin{aligned}
\gamma &= \frac{\left(x + \alpha_p\Delta x^p\right)\left(s + \alpha_d\Delta s^p\right) + \left(w + \alpha_p\Delta w^p\right)\left(z + \alpha_d\Delta z^p\right)}{2n} \\
\mu &= \sigma\gamma, \qquad \sigma = 0.1
\end{aligned}
\tag{35}
$$

Other methods[11,16,18,19] can reduce μ faster when closer to the optimal solution, such as (18), and are more useful when higher accuracy is necessary. Although (18) was found to speed up convergence for the primal-dual algorithm, it did not yield any improvement for predictor-corrector path-following.

The corrector step is taken on the condition that the smallest step size, as determined for the predictor step, is less than one.[16] This allows for large steps when far from the

boundary. A step size of less than one means that the current point is too near a boundary to take a large affine step. More centering is required in this case so a centering or corrector step is taken. The corrector step utilizes the remaining part of (32). In fact, an identical routine to the predictor step (34) may be used to compute the corrector step by redefining the residuals as

$$\begin{aligned} r_c &= r_b = r_u = 0 \\ r_{xs} &= \Delta X^p \Delta S^p - \mu e \\ r_{wz} &= \Delta W^p \Delta Z^p - \mu e \end{aligned} \tag{36}$$

The 2nd order terms in the residuals promote rapid convergence and reduce overshoot by steering the updates toward the central path trajectory. A new step size can be determined as before, but using the combined updates of (33).

Implementation Issues

Each of the algorithms discussed so far have a common enemy, namely, finite precision of computers. As the iterates come closer and closer to a boundary, they either approach zero or their dual approaches zero. In either case, the numerical conditioning degrades. In an attempt to shorten number of iterations, some algorithms use the previous solution in a trajectory of commands as the starting point for the next solution. However, for good convergence of interior-point methods, it is important that the estimate becomes feasible before it becomes complimentary, making warm starting techniques impractical. Since the algorithms are designed so that convergence of the variables is simultaneous, many singularities are likely to occur together. There are three implementation choices that help combat these numerical difficulties; a numerically

stable step direction, Cholesky factorization, and applying limits to reciprocals. Fortunately, these improvements are not difficult.

Numerically Stable Step Direction

The step Δs is traditionally defined as $\Delta s = -A^T \Delta\lambda - r_c$ but our experience has been that the calculation

$$\Delta s = -X^{-1} r_{xs} - X^{-1} S \Delta x \tag{37}$$

is numerically more stable and should replace the corresponding updates in (22) and (34).

Cholesky Factorization

Common to every interior-point algorithm is the term $\left(ADA^T\right)^{-1}$. When computing $\left(ADA^T\right)^{-1}$, it is advisable to use a stable inversion routine. Most interior-point methods employ Cholesky factorization to accomplish this. Since *D* is positive definite, ADA^T is symmetric positive definite (assuming full row rank) and can be factored as

$$ADA^T = LL^T \tag{38}$$

where *L* is an upper right triangular matrix. Once in this form, λ can be computed very easily and with more numerical stability. There are other, more advanced techniques, such as pivot ordering and Cholesky-infinity factorization that handles positive semi-definite matrices.[16] These have not been found to be beneficial to control allocation problems, presumably because they are small and dense.

Limits on Reciprocals

Inverses of the primal variables, x and w, and the diagonal matrix, D, are computed at each iteration. As the optimum point is approached, elements of both x and w can approach zero, implying that their reciprocal approaches infinity. Complementarity of the primal and dual variables also drives many of the diagonal elements of D to either zero or infinity. To prevent an internal error in the computer, each value is checked after the reciprocal is performed. The following operation was found to apply an acceptable upper limit for each reciprocal element

$$x_i^{-1} = \min\left(x_i^{-1}, 10^8\right) \quad i = 1 \ldots n \tag{39}$$

AIRCRAFT SIMULATION MODELS

Aircraft models are nonlinear in general. However, at a given flight condition, the model may be linearized. The aircraft aerodynamic model is a function of altitude, Mach number, angle-of-attack, and sideslip. Additionally, the control moments will be a function of the flight condition and the current control effector positions. Flight computers typically store multiple linear models on a grid over the entire flight envelope of the aircraft. The tests in this paper have been restricted to one flight condition for each of two aircraft. The aircraft models represent the Boeing C-17 cargo jet and an advanced tailless fighter studied by Lockheed-Martin. These aircraft were chosen because they have large numbers of effectors and because they are representative of two types of systems. The tailless fighter is an example of a system with coplanar controls,[20] whereas the C-17 does not have this property.

Given a_d, $u_{\min}$, $u_{\max}$, and CB

Convert the control allocation problem to a linear program, see (7) or (9)

Choose values for parameters ρ and ε_s

Compute starting point, $(x,\ s,\ w,\ z,\ \text{and}\ \lambda)$ see (28)-(31) and Table 1

Compute complementarity gap, $\mu = \min(0.1, 10\gamma)\gamma,\ \text{where}\ \gamma = \dfrac{x^T s + w^T z}{2n}$

Compute feasibility residuals $(r_c,\ r_b,\ r_{xs},\ r_{wz},\ \text{and}\ r_u)$ see (21) and (25)

While (24) is false

Solve for the step direction,

$$\begin{bmatrix} \Delta\lambda \\ \Delta x \\ \Delta s \\ \Delta w \\ \Delta z \end{bmatrix} = \begin{bmatrix} \left(ADA^T\right)^{-1}\left(-r_b + ADr_r\right) \\ D\left(A^T\Delta\lambda - r_r\right) \\ -X^{-1}r_{xs} - X^{-1}S\Delta x \\ -\Delta x \\ -W^{-1}Z\Delta w - W^{-1}r_{wz} \end{bmatrix} \quad \text{where} \quad \begin{aligned} D &= \left(X^{-1}S + W^{-1}Z\right)^{-1} \\ r_r &= -r_c + X^{-1}r_{xs} - W^{-1}r_{wz} + W^{-1}Zr_u \end{aligned}$$

Compute step sizes for primal and dual variables

$$\alpha_P = \min\left\{\left[-\frac{x_i}{\Delta x_i},\ 1\right] \middle| \Delta x_i < 0,\ i = 1...n\right\},\ \alpha_D = \min\left\{\left[-\frac{s_i}{\Delta s_i},\ 1\right] \middle| \Delta s_i < 0,\ i = 1...n\right\}$$

Update the variables $x,\ s,\ w,\ z,\ \text{and}\ \lambda$

$$x = x + \rho\alpha_P\Delta x \qquad s = s + \rho\alpha_D\Delta s$$
$$w = w + \rho\alpha_P\Delta w \qquad z = z + \rho\alpha_D\Delta z$$
$$\lambda = \lambda + \rho\alpha_D\Delta\lambda$$

Compute complementarity gap

Compute feasibility residuals

end while

Compute control vector, see (8) or (10) as appropriate.

Figure 1. Steps of a primal-dual interior-point algorithm.

C-17 Model

The C-17 model includes sixteen separately controlled surfaces: four elevators, two ailerons, two rudders, and eight spoilers. The flight condition for this model is at an altitude of 30,000ft and at a speed of Mach 0.69. The control effectiveness matrix is given by

$$CB = \begin{bmatrix} -0.5158 & -0.8129 & -0.8129 & -0.5158 & -0.1224 & -0.1224 & 0.0114 & 0.0169 \\ 0.2077 & 0.1126 & -0.1126 & -0.2077 & 0.4812 & -0.4812 & 0.3044 & 0.3137 \\ 0.0188 & 0.0102 & -0.0102 & -0.0188 & 0.0241 & -0.0241 & -0.3428 & -0.5118 \end{bmatrix}$$

$$\begin{bmatrix} -0.0361 & -0.0116 & 0.0192 & 0.0338 & -0.0361 & -0.0116 & 0.0192 & 0.0338 \\ -0.1056 & -0.2130 & -0.2723 & -0.3045 & 0.1056 & 0.2130 & 0.2723 & 0.3045 \\ -0.0465 & -0.0460 & -0.0457 & 0.0397 & 0.0465 & 0.0460 & 0.0457 & 0.0397 \end{bmatrix}$$

The actuator commands are given in degrees and have the following limits:

$$u_{\max} = \begin{bmatrix} 15 & 15 & 15 & 15 & 26 & 26 & 5 & 5 & 30 & 30 & 30 & 30 & 30 & 30 & 30 & 30 \end{bmatrix}^T$$

$$u_{\min} = \begin{bmatrix} -15 & -15 & -15 & -15 & -39 & -39 & -5 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

The ordering of the controls is as follows: outer left elevator, inner left elevator, inner right elevator, outer right elevator, left aileron, right aileron, lower rudder, upper rudder, four left spoilers, four right spoilers. The output vector of this model consists of pitch, roll, and yaw rates.

Tailless Model

The advanced tailless fighter model[21] includes eleven separately controlled surfaces consisting of elevons, pitch flaps, thrust vectoring, outboard leading edge flaps, spoiler

slot deflectors and all-moving tips. The flight condition for this model is 15,000ft at Mach 0.4. The control effectiveness matrix is given by

$$CB = \begin{bmatrix} -2.5114 & -2.5115 & -1.9042 & -0.9494 & -0.9494 \\ 3.7450 & -3.7450 & 0 & 1.7718 & -1.7718 \\ -0.5319 & 0.5319 & 0 & -0.4893 & 0.4893 \end{bmatrix}$$

$$\begin{bmatrix} -1.1329 & 0 & 1.5046 & 1.5046 & -0.0003 & -0.0004 \\ 0 & -0.0455 & -2.0751 & 2.0752 & -0.2887 & 0.2887 \\ 0 & -0.8174 & 0.2875 & -0.2875 & -0.1418 & -0.1418 \end{bmatrix}$$

The actuator commands are given in degrees and have the following limits:

$$u_{\max} = \begin{bmatrix} 30 & 30 & 30 & 60 & 60 & 10.61 & 10.61 & 60 & 60 & 40 & 40 \end{bmatrix}^T$$
$$u_{\min} = \begin{bmatrix} -30 & -30 & -30 & -60 & -60 & -10.61 & -10.61 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

The ordering of the controls is as follows: left and right elevons, pitch flaps, left and right all-moving tips, pitch thrust vectoring, yaw thrust vectoring, left and right spoilers slots, left and right outboard leading edge flaps. The output vector is composed of modified rotational rates. Specifically, the components are pitch rate, stability axis roll rate, and a blend of sideslip and stability axis yaw rate.

TEST RESULTS AND COMPARISONS

A set of uniformly distributed random vectors are used to test each algorithm, formulation, and aircraft model to provide comparisons. The vectors are angular acceleration commands for the aircraft that are to be generated by the deflecting control surfaces. The set, Ψ, is comprised of three subsets of vectors, namely, achievable, exactly achievable, and unachievable acceleration commands, $\Psi = \left\{ \frac{1}{2}\Omega, \Omega, 2\Omega \right\}$. The

subset, $\Omega \subset \Re^3$, consists of 1000 vectors in random directions that are scaled so that each vector just touches the boundary of the attainable set. Two more subsets of vectors are then created by halving and doubling the magnitudes. Most control allocation algorithms behave differently for these three subsets of commands. The test set was therefore created this way to ensure that each case was adequately represented.

Comparisons of the algorithms are made regarding accuracy and control effort. Simplex solutions are provided as baselines since their solutions are exact (i.e. within machine precision). The weighting of the control term for the baseline solutions was chosen to be $h = 1\text{x}10^{-6}$. The selection of h is somewhat arbitrary, but must be large enough for numerical reliability yet small enough to keep the control term secondary. Conclusions from the simulations regarding stopping criteria, ε_s, and the weighting parameter, h, in the mixed ℓ_1-norm optimization problem are determined and used to evaluate the computational efficiency of the algorithms.

It turns out that the behavior of the primal-dual and predictor-corrector methods are nearly identical for the following tests. Indeed, the results for each aircraft model are similar as well. In light of this, except where otherwise noted, the plots in this paper specifically use the C-17 model and the primal-dual algorithm, but are also representative of each combination of interior-point algorithm and model.

Accuracy

To determine the accuracy of the interior-point algorithms, measures for the acceleration accuracy and control effort were evaluated. These characteristics were treated separately so that specific behavior could be observed. For a desired

acceleration, a_d, and the baseline command (obtained from a simplex algorithm), u_s, the accuracy of an interior-point solution, *u*, was measured by

$$\varepsilon_a = \frac{\left\|a_d - CBu\right\|_1 - \left\|a_d - CBu_s\right\|_1}{\left\|CB\right\|_1} \tag{40}$$

The error is normalized by the ℓ_1-norm of control effectiveness matrix, $\left\|CB\right\|_1 = \max_j \sum_{i=1}^{n} \left|CB_{ij}\right|$. This definition of ε_a gives it an interpretation in terms of an angle difference of the effectors using the interior-point method versus the simplex method. This angle can then be compared to the resolution of the control effectors. An acceptable error is considered to be $\varepsilon_a \le 0.1$ degrees.

Control effort is again measured by the difference between the interior-point solution and the simplex solution. Multiple solutions often exist which result in identical *CBu*'s, yet very different *u*'s. Considering a preferred control, u_0, the measure of control effort, ε_u, was chosen to be

$$\varepsilon_u = \left\|u - u_0\right\|_1 - \left\|u_s - u_0\right\|_1 \tag{41}$$

If ε_u is negative, the interpretation is that the interior-point solution is closer to the preferred control than the simplex solution. For this to be possible, the optimal point must have multiple solutions. One such case is the tailless aircraft model with the direct allocation approach. In this case, the simplex method usually finds solutions with more controls at their limits than the interior-point algorithms, resulting in a higher energy expense.

This section shows the behavior of the different algorithms and formulations as functions of the stopping tolerance, ε_s, and the control weighting, *h*. This insight assists

the designer in choosing appropriate values for these parameters. In the selection of ε_s and h, the metrics ε_a and ε_u are the chief concern, but the computational burden that is related to ε_s must also be considered. Choosing the largest value of ε_s that still satisfies one's requirements will likely result in the smallest number of computations.

Direct Allocation Objective

Acceleration Accuracy

Figure 2 is a comparison of the mean acceleration error as calculated in (40) for the direct allocation objective. Tests for the C-17 model are plotted versus the tolerance, ε_s, used in the respective stopping criteria for the three interior-point algorithms. Both path-following methods demonstrate similar abilities in converging to very high accuracy.

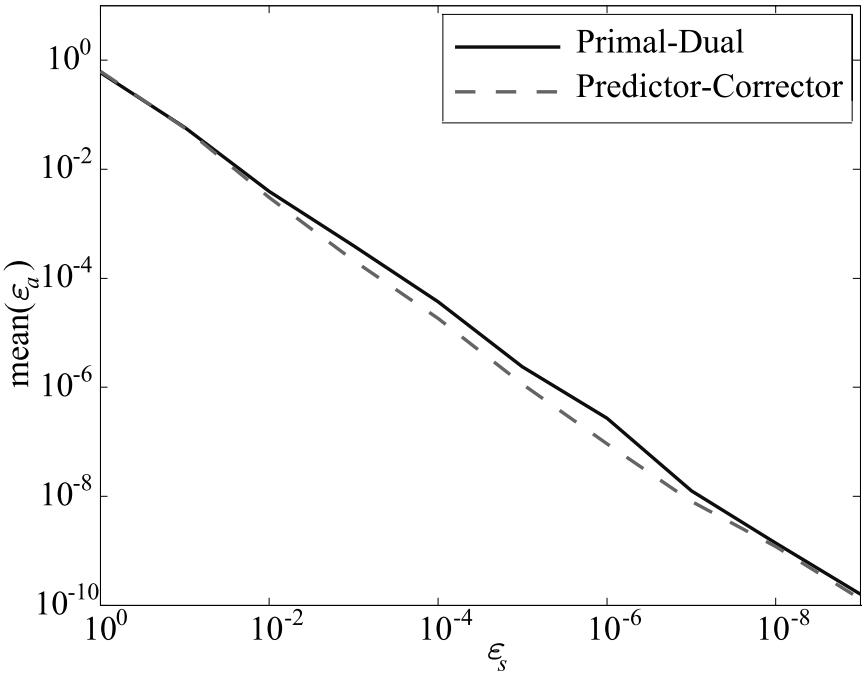

Figure 2. Mean acceleration error for the direct allocation formulation. (C-17)

The acceleration error can be thought of as the deviation of u from u_s. In that sense, it can be measured in degrees, *i.e.* the difference in angle between the actual command and the baseline command. By analyzing the figures, an appropriate tolerance may be selected for the desired accuracy. For example, assuming a desired accuracy of $\varepsilon_a \le 0.1$ for the tailless fighter, one should choose $\varepsilon_s \le 0.1$ for either interior-point method.

Control Accuracy

The direct allocation method does not explicitly minimize the control cost because it is not designed to. However, for systems with coplanar controls (*i.e.* three or more controls are linearly dependent), such as the tailless fighter, the direct allocation problem does not have a unique solution. The simplex method arrives at the optimal vertex which always has variables at their limits, often maximizing the control. Interior-point methods, on the other hand, travel from the interior, and typically converge to an optimal solution with smaller variables. This inherent trait often results in solutions that have fewer saturated controls than that of the simplex method. For example, the maximally attainable acceleration $a_d = \begin{bmatrix} 133.5289 & -349.2778 & 37.8923 \end{bmatrix}^T$ for the tailless model has a simplex solution, u_s, and primal-dual interior-point solution, u, equal to

$$u_s = \begin{bmatrix} -30.0 & 30.0 & 30.0000 & -60.0000 & -53.7145 & 6.6754 & 10.61 & 60.0 & 0.0 & 0.0 & 40.0 \end{bmatrix}^T$$
$$u = \begin{bmatrix} -30.0 & 30.0 & -6.7738 & -19.1244 & -12.8389 & -0.024 & 10.61 & 60.0 & 0.0 & 0.0 & 40.0 \end{bmatrix}^T$$

The solutions are identical except for the 3rd thru 6th components. The acceleration error is $\varepsilon_a = 1.7585\text{x}10^{-5}$. Clearly the interior-point solution is much smaller in magnitude than the simplex solution even though practically identical accelerations are generated.

The example given has about 27% smaller control than the simplex control, but the average percentage difference using an interior-point method was found to be about 5% smaller. Figure 3 is a histogram showing the distribution of the difference (as a percentage) in control costs between the simplex and primal-dual solutions for the tailless fighter.

Mixed ℓ_1-Norm Objective

Visualization of the errors of the mixed ℓ_1-norm is best done with a three dimensional surface so that the relationship of both the weighting, *h*, and the stopping rule tolerance, ε_s, can be simultaneously observed. The error is computed from the simplex solution using $h = 10^{-6}$. Robust selection of *h* and ε_s for each algorithm can be done by using values that satisfy the $\varepsilon_a \leq 0.1$ selected tolerance.

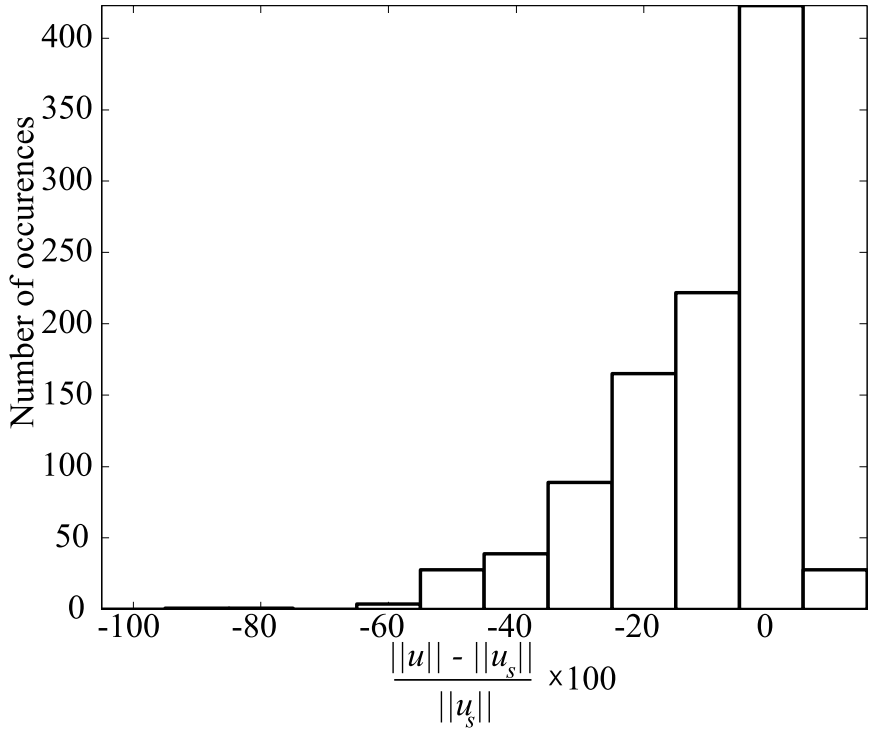

Figure 3. Histogram of percent difference in control costs.

Acceleration Accuracy

Figure 4 shows the mean error surface. The highest accuracy is clearly at or near the point with the smallest tolerance and where *h* is the same as that of the baseline method ($h = 1\text{x}10^{-6}$). One finds that suitable values for the path-following methods are, on average, $\varepsilon_s \leq 0.1$ for either model and with $h \leq 0.01$ and $h \leq 0.001$ for the C-17 and tailless fighter, respectively. An empirically derived rule of thumb for acceleration accuracy was found to be $\varepsilon_a \leq 100\varepsilon_s$.

Control Cost

The control effort depends heavily on the penalty placed on the control term in the objective function. A high value of *h* results in solutions that favor smaller magnitude control values at the expense of the acceleration error. Figure 5 shows the surfaces for the control costs using the interior-point methods. Again, the results between both algorithms are essentially the same. In fact, we find that the control cost, which is less than or equal to zero for $h \leq 10^{-7}$ and $\varepsilon_s \leq 10h$ is also the same for either aircraft model.

Selection of h and ε_s

By analyzing both plots of acceleration error and control cost, parameters *h* and ε_s can be judiciously selected. Table 2 lists these parameters so that one set can be chosen to satisfy both models. The stopping tolerance should be chosen as large as possible to minimize iteration count. The control cost weighting should also be chosen as large as allowed for minimum control cost. From the table, acceptable values are $h = 0.001$ and $\varepsilon_s = 0.01$.

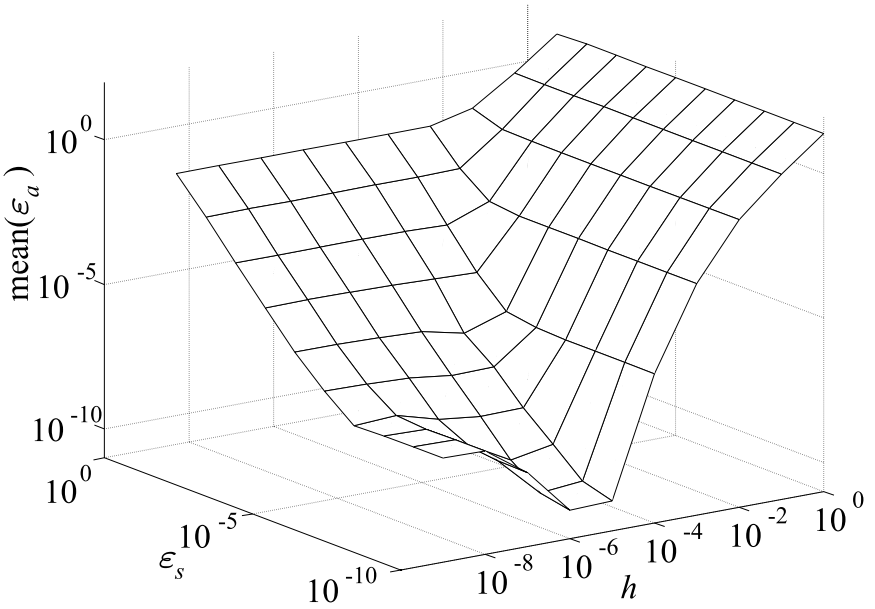

Figure 4. Mean acceleration error for mixed optimization

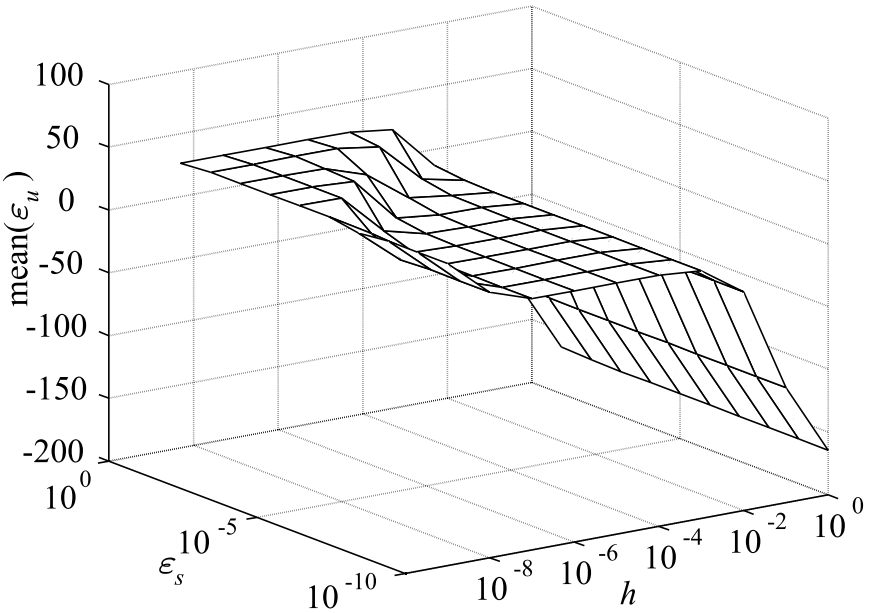

Figure 5. Control cost using mixed optimization.

Table 2. Parameter selection for the path-following methods.

	h	ε_s
C-17 $\varepsilon_a \le 0.1$ Tailless $\varepsilon_a \le 0.1$	≤ 0.001 ≤ 0.01	≤ 0.1
C-17 $\varepsilon_u \le 0$ Tailless $\varepsilon_u \le 0$	$\ge 10^{-7}$	$\le 10h$
$\varepsilon_a \le 0.1$ and $\varepsilon_u \le 0$	$10^{-7} \le h \le 0.001$	$\varepsilon_s \le 10h$

Computational Cost

Computational efficiency is extremely important for real-time applications. The computational efficiency of each algorithm is measured here by the number of iterations and by the number of floating point operations, or flops (using the Matlab 5.3 *flops* function), for platform independence.

Iteration Count and Floating Point Operations

Iteration count is usually the parameter of interest for analyzing the speed of linear programming algorithms. For small problems, such as found in control allocation, however, the number of floating point operations also offers valuable insight. The models and formulations can be converted to a problem size to analyze the algorithm efficiency. Problem size is typically defined by

$$L = mn + m + n \tag{42}$$

with m and n representing the numbers of rows and columns of A. The number of flops per iteration count is dependent on the sizes of A, b, and c, which correspond to the three terms in (42). Figure 6 and Figure 7 display the iteration count and the number of

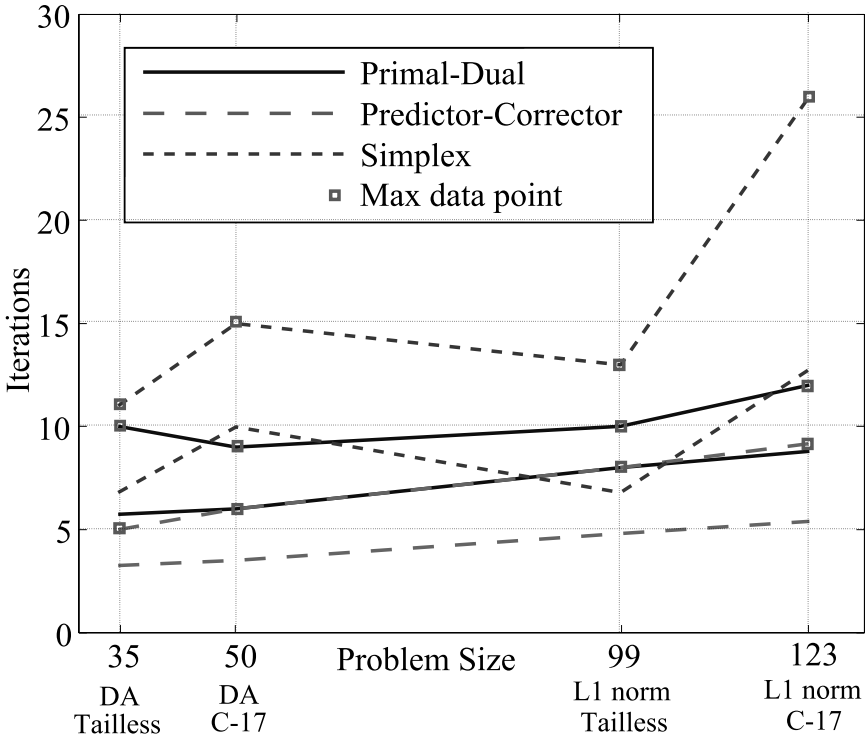

Figure 6. Maximum and mean number of iterations.

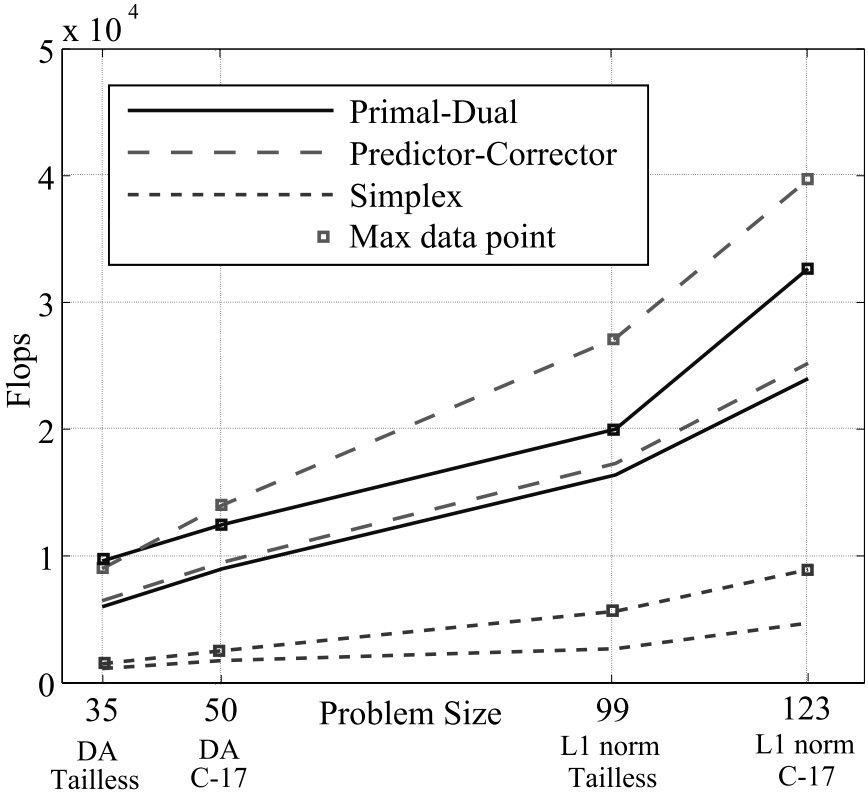

Figure 7. Maximum and mean number of flops.

flops, respectively, required to converge using acceptable parameters just determined ($h = 0.001$ and $\varepsilon_s = 0.01$). The first graph plots the number of iterations vs. problem size. The number of iterations is shown for each algorithm using the same line pattern for both mean and maximum except that the maximum line has squares marking the data points.

From Figure 6 it may appear that the simplex is the slowest algorithm followed by the primal-dual followed by the predictor-corrector. But in fact the reverse is true. The number of iterations does not tell the whole story because the predictor-corrector method requires about 1.4 times the number of flops per iteration as compared to the primal-dual method. The mean values of each interior-point algorithm are very similar. Predictor-corrector requires the fewest number of iterations, but the most flops per iteration, which ultimately places it at a higher computational need than the primal-dual interior-point method when the worst case is considered. The worst case is important for control applications to determine whether an algorithm will converge within the prescribed cycle time.

Convergence Behavior

One of the most attractive traits of interior-point algorithms is their uniform convergence towards the optimum. Figure 8 and Figure 9 show the average convergence of the acceleration error as a function of number of iterations with a mixed ℓ_1-norm formulation for the C-17 and tailless fighter aircraft models, respectively. The weighting factor, *h*, is set very small (10^{-9}) so that the acceleration error completely dominates the cost. In the figures, the first iteration is taken as the starting point calculation. Although

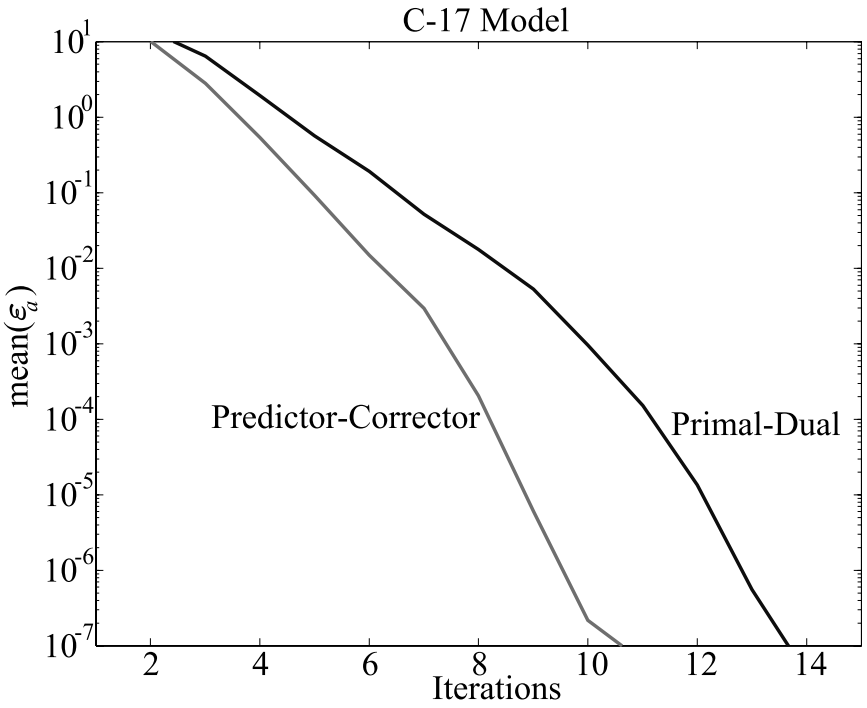

Figure 8. Average convergence of interior-point methods.

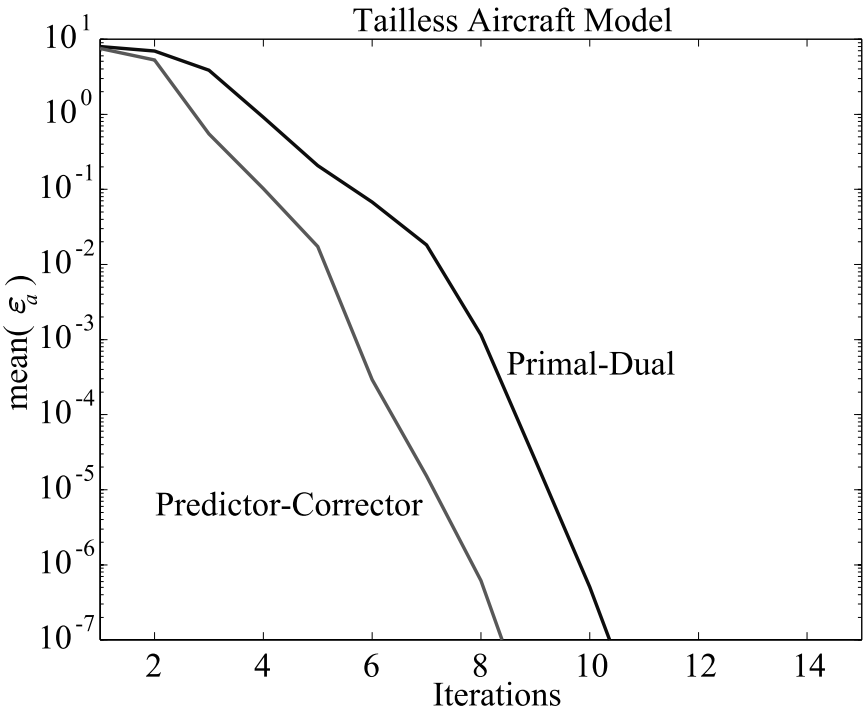

Figure 9. Average convergence of interior-point algorithms.

the algorithms converge at different rates, they all show consistent progress towards an optimal solution. As opposed to other techniques, interior-point methods are well-suited to implementations with fixed number of iterations.

Real-Time Data

The primal-dual algorithm using the mixed ℓ_1-norm objective was implemented in the C programming language to get an idea of the real-time requirements for these methods. The predictor-corrector method was not implemented, but its timing can be extrapolated from the flops and iteration comparison shown. Timing is consistent for each iteration and does not vary. Table 3 shows the timing data obtained using a PC based on a 1.46GHz AMD Athlon 1700 processor. Convergence data is given for the set Ψ using a weighting parameter of $h = 10^{-6}$ and a stopping tolerance at $\varepsilon_s = 10^{-6}$.

The data show that the maximum time required to compute accurate solutions using an interior-point algorithm is less than 200μs for either aircraft model. Although today's flight computers are about a decade behind current computers, this experiment shows

Table 3. Timing and convergence data using data set Ψ.

$h=10^{-6}$, $\varepsilon_s=10^{-6}$	Mean ε_a	Max ε_a	Mean iterations	Max iterations	Max convergence time
Primal-dual Tailless	$1.4\text{x}10^{-6}$	$1.5\ \text{x}10^{-3}$	12	22	150μs
Primal-dual C-17	$3.1\ \text{x}10^{-7}$	$5.9\ \text{x}10^{-5}$	12	21	170 μs
Simplex Tailless	$5.9\ \text{x}10^{-14}$	$8.0\ \text{x}10^{-13}$	10	17	41μs
Simplex C-17	$1.8\ \text{x}10^{-14}$	$3.8\ \text{x}10^{-14}$	15	25	74μs

that interior-point algorithms are now, or in the near future, within our ability to be implemented in real-time control allocation applications. Although there appears to be a large difference in accuracy, the values are much smaller than resolutions of real effectors which make the difference negligible.

CONCLUSIONS

The control allocation problem has been posed here with two different objectives, namely, the direct allocation objective and a mixed ℓ_1-norm objective. These objectives were transformed into efficient linear programming formulations so that they could be solved using any linear programming algorithm. Presentations of two common interior-point algorithms, primal-dual path-following, and predictor-corrector path-following were given. Adaptations to the starting points and the stopping rules were included in the descriptions. The increase in the number of computations due to infeasible starting points and two-sided bounds was insignificant. Implementation techniques such as Cholesky factorization and placing limits on variables were found to improve numerical stability. Tests using the control allocation algorithms were performed on linear models of two different aircraft, a tailless fighter and a C-17. Both interior-point algorithms performed well. The primal-dual path-following method was found to be slightly better overall than the predictor-corrector path-following method, as it required fewer computations and was simpler to implement. However, we suspect that the predictor-corrector method may be better for larger scale problems. Guidelines for choosing the stopping rule tolerance and the control weighting for the mixed ℓ_1-norm formulation were given, and specific values for the two aircraft models were found.

In summary, interior-point algorithms can be successfully applied to control allocation methods. Although they are slower than the simplex algorithm, interior-point algorithms progress uniformly towards the optimum, so that implementations with a fixed number of iterations can be considered. Interior-point methods are well behaved for large scale systems and one may expect good scalability for problems with many effectors as well as more competitive performance with simplex methods. Interior-point algorithms can also be applied to nonlinear programming problems so that extensions to control allocation problems with nonlinear cost functions or with moments that are nonlinear functions of the effectors are possible.

The sensitivity of the solution is an area of interest for future research. For problems formulated using direct allocation criteria we suspect that interior-point methods will be less sensitive than simplex methods due to the gradient nature of the algorithm. However, for problems cast in the ℓ_1-norm format the sensitivity is independent of the method used. The ℓ_1-norm solution without a weight on the control vector is more likely to be sensitive since the control may jump by a finite amount for an arbitrarily small benefit in performance. A change in the solution will only occur for improvements that are greater than a value determined by the weighting factor.

REFERENCES

[1]Durham,W.C., "Constrained Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993, pp. 717-725.

[2]Durham,W.C., "Constrained Control Allocation: Three-Moment Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 2, 1994, pp. 330-336.

[3]Durham,W.C., "Attainable Moments for the Constrained Control Allocation Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 6, 1994, pp. 1371-1373.

[4]Bodson, M., "Evaluation of Optimization Methods for Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 703-711.

[5]Enns, D., "Control Allocation Approaches," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* AIAA Paper 98-4109, Aug. 1998.

[6]Buffington, J., "Modular Control Law Design for the Innovative Control Effectors (ICE) Tailless Fighter Aircraft Configuration 101-3," Air Force Research Laboratory, AFRL-VA-WP-TR-1999-3057, Wright Patterson AFB, OH, May 1999.

[7]Bodson, M. and Pohlchuck, W., "Command Limiting in Reconfigurable Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 4, July-August 1998, pp. 639-646.

[8]Buffington, J., Chandler, P., and Pachter, M., "Integration of On-line System Identification and Optimization-based Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* AIAA Paper 98-4487, Aug. 1998.

[9]Nelson, M.D. and Durham, W.C., "A Comparison of Two Methods Used to Deal with Saturation of Multiple, Redundant Aircraft Control Effectors," *AIAA Atomospheric Flight Mechanics Conference and Exhibit*, AIAA Paper 2002-4498, 2002.

[10]Megiddo, N., "Pathways to the Optimal Set in Linear Programming," *Progress in Mathematical Programming: Interior-Point and Related Methods*, edited by N. Megiddo, Springer-Verlag, NY, NY, 1989, pp. 131-158.

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

[12]Lustig, I.J., Marsten, R.E., and Shanno, D.F., "On Implementing Mehrotra's Predictor-Corrector Interior-Point Method for Linear Programming, *SIAM Journal on Optimization*, Vol. 2, No. 3, Aug. 1992, pp. 435-449.

[13]Dantzig, George B., *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963, p. 24.

[14]Karush, W., "Minima of functions of several variables with inequalities as side constraints," MS thesis, University of Chicago, 1939.

[15]Kuhn, H. and Tucker, A.W., "Nonlinear programming," *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, edited by J. Neyman, University of California, Berkley, CA, 1951.

[16]Zhang, Y., "Solving Large-Scale Linear Programs by Interior-Point Methods Under the Matlab Environment," Department of Mathematics and Statistics, University of Maryland, TR96-01, Baltimore, Feb. 1996.

[17]Kojima, M., Megiddo, N., & Mizuno, S., "A Primal-Dual Infeasible-Interior-Point Algorithm for Linear Programming," *Mathematical Programming*, Vol. 61, 1993, pp. 263-280.

[18]Gondzio, J. and Terlaky, T., "A Computational View of Interior-Point Methods," *Advances in Linear and Integer Programming*, edited by J. Beasley, Clarendon Press, Oxford, 1996, pp. 103-144.

[19]Wright, S., *Primal-Dual Interior-Point Algorithms,* SIAM Publications, Argonne National Laboratory, Argonne, Ill., 1997, p. 207.

[20]Petersen, J. and Bodson, M., "Fast Implementation of Direct Allocation with Extension to Coplanar Controls," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 3, 2002, pp. 464-473.

[21]Buffington, J., "Tailless Aircraft Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* AIAA Paper 97-3605, Aug. 1997.