2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

On Active Set Algorithms for Solving Bound-Constrained Least Squares Control Allocation Problems

Brad Schofield

Abstract— **Control allocation problems, in which optimal actuator values are assigned based on desired control actions and actuator constraints, are typically formulated as constrained optimization problems. Initially, only approximate solutions to the problems were deemed tractable for real-time applications. Recently however, active set algorithms have been identified as a means for solving control allocation problems in real-time applications. In this paper a modified active set algorithm for the solution of bound-constrained least-squares problems is presented, which has numerous advantages particularly for time-varying control allocation problems. A vehicle dynamics control system for rollover mitigation is used as an example.**

I. INTRODUCTION

In recent years considerable research has been done in the area of control allocation, primarily in the context of aerospace and marine applications [1], [2], [3], but also for road vehicles [4], [5], [6], [7]. Control allocation refers to the process of mapping a given control action to individual allocator inputs, typically in the case of overactuated systems where there are more actuators than desired control actions. A widely used approach is to set up an optimization problem which is then solved at each sample time. Previously, solving such optimization problems in the presence of constraints was not regarded as being tractable for real-time applications, and a number of approximate methods were developed [8]. More recently, exact solution of the control allocation problem through the use of active set methods has been investigated [9]. These methods apply to bound-constrained quadratic programming problems (BCQP) of the form:

$$\begin{array}{rl} \min & q(u) \\ \text{subject to} & \underline{u} \leq u \leq \overline{u} \end{array} \tag{1}$$

where $\underline{u}$ and $\overline{u}$ are the lower and upper bounds on the control input u respectively, and $q(u)$ is a convex quadratic cost function. A large number of control allocation problems involving actuator constraints may be written on this form.

As an alternative to calculating the solution to such problems online, methods exist in which an offline solution is obtained. An example of such a method is multi-parametric quadratic programming (MPQP) [5]. MPQP is computationally efficient, but requires a considerable amount of memory to store the solution. A drawback of all methods which compute a solution offline is that it is difficult to handle changes in the control allocation problem. The application used in this paper is an example of a problem in which the control allocation problem is parameter-varying.

Brad Schofield is with the Department of Automatic Control, Faculty of Engineering, Lund University, SE-221 00 Lund, Sweden. brad.schofield@control.lth.se

Classical active set algorithms typically only make one change to the working set each iteration, which can result in a large number of iterations if there are many variables, or if the initial working set was very different from the optimal active set [10]. For large scale problems this is a major issue and research has been done to find algorithms that are capable of identifying the optimal active set more quickly. This is often done using gradient projection methods [11]. For small scale systems, this problem has received less attention. The use of these algorithms for the solution of control allocation problems in real-time does however raise the issue of computation time even for small scale systems. The performance of classical active set algorithms in real-time settings such as model predictive control and control allocation is greatly improved by ability to re-use the optimal solution and active set obtained in the previous sampling instant as starting points. This is known as 'hotstarting'. However, when the allocation problem is time-varying, the use of hotstarting can cause problems, as discussed in [7]. This is particularly true when time-varying constraints (such as rate constraints) are present. The algorithm proposed in this paper is more efficient at finding the optimal active set, and performs well even without hotstarting.

II. CONTROL ALLOCATION

For model-based control design, it is often easier to work with models describing the response of the system to external forces and moments, rather than actuator positions. An example of this is vehicle dynamics control, in which the dynamic model of the vehicle uses resultant forces and moments as inputs, rather than the actual actuator inputs. This is not only more intuitive from the point of view of modeling and control, but it is also generally true that there are more actuators than there are resultant forces and moments. Similar examples can be found in aerospace and marine vehicle control. In control allocation, the control design task is effectively split into two steps. In the first step, standard control design methods are used to obtain 'virtual' control signals. The second step consists of transforming these virtual control signals into 'actual' control signals which are applied to the process. This is illustrated in Figure 1, in which the controller generates the virtual controls v, which are transformed by the control allocator into actual controls u.

Generally, the relationship between virtual and actual controls is $v(t) = g(u(t))$ where $v(t) \in \mathbb{R}^k$ are the virtual controls, $u(t) \in \mathbb{R}^m$ are the actual controls and $g : \mathbb{R}^m \to \mathbb{R}^k$ is the mapping from actual to virtual controls, where

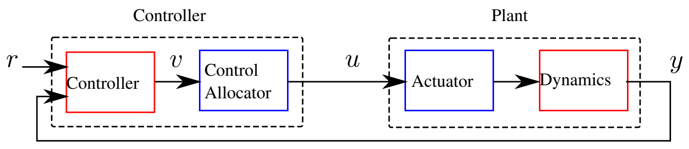

Fig. 1. Control system structure with control allocation.

$m > k$. The majority of the literature deals with the linear case, where the actual and virtual controls are related by a *control effectiveness matrix* B:

$$v(t) = Bu(t) \tag{2}$$

A common approach to control allocation is to formulate an optimization problem in which the allocation error $||Bu(t)-v(t)||_2$ is minimized, subject to actuator constraints. In the presence of bound-type actuator constraints, a linearly-constrained quadratic programming problem may be formulated. Such problems can take the form:

$$\begin{aligned} u &= \arg\min_{u\in\Omega} ||W_u(u-u_d)||_2 \\ \Omega &= \arg\min_{\underline{u}\le u\le\overline{u}} ||W_v(Bu-v)||_2 \end{aligned} \tag{3}$$

where W_u and W_v are diagonal weighting matrices, u_d is some desired actual control value, and $\underline{u}$ and $\overline{u}$ are constraints on the actual controls. This type of problem is known as Sequential Least-Squares (SLS), since the solution is computed in two steps. First, the weighted allocation error $||W_v(Bu-v)||$ is minimized. If feasible solutions are found, the 'best' solution is obtained by minimizing $||W_u(u-u_d)||$. A faster algorithm can be obtained by approximating the SLS formulation as a Weighted Least-Squares (WLS) problem:

$$u = \arg\min_{\underline{u}\le u\le\overline{u}} \left(||W_u(u-u_d)||_2^2 + \gamma||W_v(Bu-v)||_2^2\right) \tag{4}$$

The parameter γ is typically chosen to be very large in order to emphasize the importance of minimizing the allocation error. The cost function may be rewritten as:

$$\begin{aligned} &||W_u(u-u_d)||_2^2 + \gamma||W_v(Bu-v)||_2^2 \\ &= \left\| \underbrace{\begin{pmatrix} \gamma^{\frac{1}{2}}W_vB \\ W_u \end{pmatrix}}_{A} u - \underbrace{\begin{pmatrix} \gamma^{\frac{1}{2}}W_vv \\ W_uu_d \end{pmatrix}}_{b} \right\|_2^2 \end{aligned} \tag{5}$$

which allows the minimization problem to be written as:

$$\begin{aligned} \min \quad & ||Au-b||_2^2 \\ \text{subject to} \quad & \underline{u} \le u \le \overline{u} \end{aligned} \tag{6}$$

This is exactly the form in (1), with $q(u) = ||Au-b||_2^2$.

III. ACTIVE SET ALGORITHMS

In Section II it was shown how a typical control allocation formulation can take the form of a bound-constrained least squares problem. In this section, the solution of such problems using active set methods will be examined. First, a general description of active set methods is given. A more specific algorithm, summarized in [10] and used in the context of control allocation in [9], is then examined.

A. Classical Primal Active Set Algorithm

In this section a general description of active set algorithms is given, following the presentation in [11]. More details can be found in [10] and [12]. This type of algorithm is sometimes referred to as the classical primal active set algorithm (CPASA).

The principal of operation of all active set methods is that at each iteration, the active inequality constraints are regarded as equality constraints, and the remaining constraints are disregarded. In the case of bound-constrained problems, an active constraint corresponds to a variable holding a constant value, which simplifies the solution since variables corresponding to the active constraints are simply removed from the resulting optimization problem. Once a minimum to this problem is found, the Karush-Kuhn-Tucker (KKT) optimality conditions are checked. If they are fulfilled, then the algorithm stops, otherwise, one of the active constraints breaking the KKT conditions is removed. For bound-constrained least squares problems, the KKT conditions are equivalent to:

$$\begin{aligned} \frac{\partial q(u_j)}{\partial u_j} &= 0, \quad \underline{u}_j < u_j < \overline{u}_j \\ \frac{\partial q(u_j)}{\partial u_j} &\ge 0, \quad u_j = \underline{u}_j \\ \frac{\partial q(u_j)}{\partial u_j} &\le 0, \quad u_j = \overline{u}_j \end{aligned} \tag{7}$$

Several definitions may be made. The *working set* $\mathcal{W}_k$ at iteration k is a subset of the *active set* $\mathcal{A}(u^k)$:

$$\mathcal{A}(u^k) = \{i : u_i = \underline{u}_i \text{ or } u_i = \overline{u}_i\}$$

Variables in $\mathcal{W}_k$ are known as *bound* variables. Variables not in $\mathcal{W}_k$ are known as *free* variables. It is also useful to define the *binding set* $\mathcal{B}(u)$ as:

$$\begin{aligned} \mathcal{B}(u) = \{i : u_i = \underline{u}_i \text{ and } \partial_i q(u) \ge 0 \text{ or } u_i = \overline{u}_i \\ \text{and } \partial_i q(u) \le 0\} \end{aligned} \tag{8}$$

The starting point u^0 of the algorithm is assumed to be feasible (such a starting point is trivial to find in the case of bound-constrained problems) and $\mathcal{W}_0 \subset \mathcal{A}(u^0)$. The next iterate u^{k+1} may be found by solving:

$$\begin{aligned} \min_p \quad & q(u^k+p) \\ \text{subject to} \quad & p_i = 0, i \in \mathcal{W}_k \end{aligned} \tag{9}$$

This corresponds to an unconstrained minimization problem in the free variables. Given the optimal perturbation p the next iterate u^{k+1} may be found. The standard method is to calculate an α:

$$\alpha = \max\{\alpha \in [0,1] : \underline{u} \le u^k + \alpha p \le \overline{u}\} \tag{10}$$

from which the next iterate is found:

$$u^{k+1} = u^k + \alpha p$$

If $\alpha = 1$, the optimality conditions are checked. Otherwise, the new active constraint is added to the working set.

B. Active Set Algorithm for Bound Constrained Least Squares

In [9] an active set algorithm for the solution of bound-constrained least squares control allocation problems was presented, based on a more general algorithm in [10]. This algorithm will be used as a benchmark for comparison with the modified algorithm, and is briefly outlined in this section. Consider the least squares problem:

$$\min_u \quad ||Au-b||_2 \tag{11a}$$

$$\text{subject to} \quad Bu = v \tag{11b}$$

$$\underbrace{\begin{pmatrix} I \\ -I \end{pmatrix}}_{C} u \geq \underbrace{\begin{pmatrix} \underline{u} \\ -\overline{u} \end{pmatrix}}_{U} \tag{11c}$$

The details of the active set algorithm used to solve the problem are given in Algorithm 1.

Algorithm 1: Classical primal active set algorithm

```
Let $u^0$ be a feasible starting point, satisfying (11c) ;
**for** $i = 0, 1, 2, \dots$ **do**
    Given suboptimal iterate $u^i$, find the optimal
    perturbation $p$, considering the inequality constraints
    in $\mathcal{W}$ as equality constraints and ignoring the
    remainder. This is done by solving:
        $\min_p ||A(u^i + p) - b||_2$
        $Bp = 0$
        $p_i = 0, \quad i \in \mathcal{W}$
    **if** $u^i + p$ *feasible* **then**
        Set $u^{i+1} = u^i + p$ ;
        Compute Lagrange multipliers as:
            $A^T(Au - b) = \begin{pmatrix} B^T & C_0^T \end{pmatrix} \begin{pmatrix} \mu \\ \lambda \end{pmatrix}$
        where $C_0$ consists of the rows of $C$
        corresponding to the constraints in the active set;
        **if** $\lambda \geq 0$ **then**
            $u^{i+1}$ is optimal solution;
            Return $u = u^{i+1}$
        **else**
            Remove constraint corresponding to most
            negative $\lambda$ from the working set $\mathcal{W}$;
    **else**
        Find $\alpha = \max\{\alpha \in [0,1] : \underline{u} \leq u^i + \alpha p \leq \overline{u}\}$
        and set $u^{i+1} = u^i + \alpha p$. Add bounding
        constraint to working set.
**end**
```

IV. MODIFICATIONS

The method of finding the next iterate described in (10) has several attractive features. Using this method guarantees that the cost function decreases at each iteration. Additionally, this method also works when equality constraints are present. Since equality constraints do not appear in the control allocation formulation however, there exists more freedom in the choice of the next iterate.

The modified algorithms for large-scale problems outlined in [11] are concerned with identifying the optimal active set quickly. In order to obtain a similar effect in the case of small scale problems, an alternative to (10) is proposed here.

The method in (10) can be interpreted as moving along a line between the current iterate and the unconstrained minimum until the boundary of the feasible set is encountered. This point becomes the next iterate, and the active constraint at this boundary is added to the working set. It is not obvious that this is the 'best' course of action in any sense. An alternative choice would be to saturate all free variables whose values at the unconstrained maximum lie outside the feasible set. This can be expressed as finding a matrix Γ:

$$\Gamma = \begin{pmatrix} \alpha^1 & 0 & \dots & 0 \\ 0 & \alpha^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha^n \end{pmatrix} \tag{12}$$

such that:

$$\alpha^i = \max\{\alpha^i \in [0,1] : \underline{u} \leq u + \alpha^i p \leq \overline{u}\} \tag{13}$$

The update is then:

$$u^{k+1} = u^k + \Gamma p \tag{14}$$

An alternative interpretation is that this method picks the point in the feasible set which is the closest in the Euclidean sense to the unconstrained minimum. Let P^f denote the unconstrained minimum of the problem in the free variables. Solve, for $u^f = \{u_i : i \notin \mathcal{W}_i\}$:

$$\begin{aligned} \min \quad & ||u^f - P^f||_2 \\ \text{subject to} \quad & \underline{u} \leq u^f \leq \overline{u} \end{aligned} \tag{15}$$

Denote the solution to this problem u^*. If P^f is feasible, then clearly $u^* = P^f$. Otherwise, one or more constraints will be active at u^*. The next iterate is obtained by setting $u_i^{k+1} = u_i^*$ for $i \notin \mathcal{W}_i$. The working set is then expanded by adding those active constraints for which the KKT conditions (7) are satisfied. This method of updating the iterate may be directly substituted in the standard CPASA algorithm. The problem to be solved is:

$$\min_u \quad ||Au-b||_2 \tag{16a}$$

$$\text{subject to} \quad \underbrace{\begin{pmatrix} I \\ -I \end{pmatrix}}_{C} u \geq \underbrace{\begin{pmatrix} \underline{u} \\ -\overline{u} \end{pmatrix}}_{U} \tag{16b}$$

This modified algorithm is outlined in Algorithm 2.

A. Convergence Properties

It will now be shown that this method of updating the iterate has attractive properties, including convergence to the optimum in a maximum of $2n-1$ steps, when starting with an empty working set. To prove this, the following lemma is needed.

Algorithm 2: Modified active set algorithm

```
Let $u^0$ be a feasible starting point, satisfying (16b) ;
for $i = 0, 1, 2, \dots$ do
    Given suboptimal iterate $u^i$, find the optimal
    perturbation $p$, considering the inequality constraints
    in $\mathcal{W}$ as equality constraints and ignoring the
    remainder. This is done by solving:
        $\min_p ||A(u^i + p) - b||_2$
        $p_i = 0, \quad i \in \mathcal{W}$
    if $u^i + p$ feasible then
        Set $u^{i+1} = u^i + p$ ;
        Compute Lagrange multipliers as:
            $A^T(Au - b) = C_0^T \lambda$
        where $C_0$ consists of the rows of $C$
        corresponding to the constraints in the active set;
        if $\lambda \geq 0$ then
            $u^{i+1}$ is optimal solution;
            Return $u = u^{i+1}$
        else
            Remove constraint corresponding to most
            negative $\lambda$ from the working set $\mathcal{W}$;
    else
        Find $\alpha^i = \max\{\alpha^i \in [0, 1] : \underline{u} \leq u + \alpha^i p \leq \overline{u}\}$
        and set $u^{k+1} = u^k + \Gamma p$;
        Compute Lagrange multipliers for the active
        constraints;
        Add constraints satisfying KKT conditions to
        working set.
end
```

Lemma 1: Let u^* denote the solution to (15) for a given set of free variables u^f associated with a problem of the form (6). Then, at least one of the constraints active at u^* will be also be active at the solution of (6).

Proof: Assume without loss of generality that $P_a^f \succeq \overline{u}_a$ for some set of indices a. This implies that $u_a^* = \overline{u}_a$. Assume now that no constraints active at u^* are active at the solution to (6), denoted u^{opt}. This implies $u_a^{opt} \succ \overline{u}_a$. Since u^{opt} must lie on the boundary of the feasible set, there must exist a separating hyperplane, passing through u^{opt}, which separates the feasible set from a level set $\mathcal{C}$, containing u^{opt}, of the original objective function in (6). Since P^f lies within $\mathcal{C}$, the line $P^f - u^{opt}$ must also lie in $\mathcal{C}$, by convexity. But since $P_a^f \succeq \overline{u}_a \succ u_a^{opt}$ there must exist a δ such that $u_a^{opt} + \delta(P_a^f - u_a^{opt})$ lies in both $\mathcal{C}$ and the feasible set. Thus no separating hyperplane exists at u^{opt}. If one or more of the constraints active at u^* were active at u^{opt}, then it is no longer generally true that $u_a^{opt} + \delta(P_a^f - u_a^{opt})$ lies within the feasible set, and a separating hyperplane may then exist. ■

This lemma formalizes the idea that the proposed updating method acts to find the optimal active set quickly. It can be thought of as identifying the constraints which are 'closest' to the unconstrained minimum of the original problem.

Proposition 1: The active set algorithm with updating as in (15) and with the initial working set empty, converges in a maximum of $2n - 1$ iterations where n is the number of optimization variables.

Proof: Lemma 1 shows that in each iteration, at least one of the constraints that become active will be active at the optimum of the original problem. Only those constraints which satisfy the KKT conditions (7) at u^* are added to the working set. The constraints satisfying the KKT conditions at u^* may not necessarily be optimal at u^{opt} however. Consider the case where n constraints are active, of which only one which remains active at u^{opt}. The worst case occurs when the $n - 1$ constraints which will *not* be active at u^{opt} fulfill the KKT conditions at u^* and are added to the working set, while the remaining constraint does not. In this case $n - 1$ future iterations will be required to remove these constraints from the working set, since only one constraint may be removed at each iteration. An additional $n - 1$ iterations are required to locate the optimum, giving a total of $2n - 1$ iterations in the worst case. ■

B. Properties relevant to Real-Time Applications

As previously mentioned, standard active set algorithms used in real-time settings benefit significantly from the use of hotstarting, where the results from the previous sampling instant are used as a starting point for the next sampling instant. Complications arise, however, when the constraints vary with time. Time varying constraints can arise when rate constraints are present [9], [7], or when the constraints depend on time varying parameters. When constraints vary from sample to sample, using the solution from the previous sample is not straightforward. For instance, the solution may no longer be feasible with respect to the new constraints, or certain constraints active at the previous solution may no longer be active. Such situations can violate the starting point assumptions of the active set algorithm and lead to incorrect behaviour. While logical checks could be carried out within the algorithm to ensure that starting conditions are met, this would influence the behaviour of the resulting algorithm. The proposed algorithm avoids this problem by removing the need for hotstarting. In terms of computational complexity, the modified algorithm adds only an addition check of the KKT conditions.

V. EXAMPLES

A. Two Dimensional Example

In order to visualize the operation of the proposed modification it is useful to examine a two-dimensional example. Consider the problem defined by:

$$A = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix}, \quad b = \begin{pmatrix} 50 \\ 50 \end{pmatrix}$$
$$\underline{u} = \begin{pmatrix} -10 \\ -10 \end{pmatrix}, \quad \overline{u} = \begin{pmatrix} 10 \\ 10 \end{pmatrix} \tag{17}$$
$$u_d = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \gamma = 1000$$

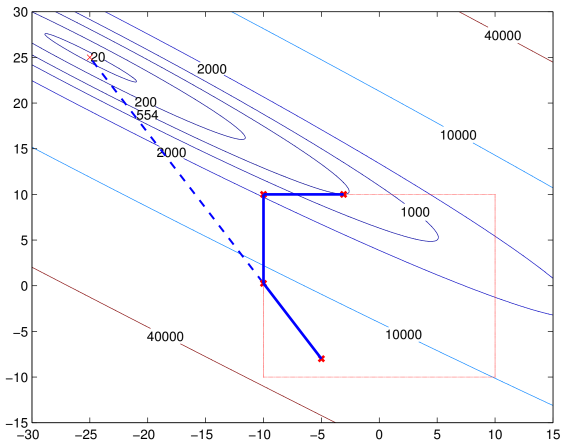
Fig. 2. The two-dimensional example (17), solved using the standard algorithm.

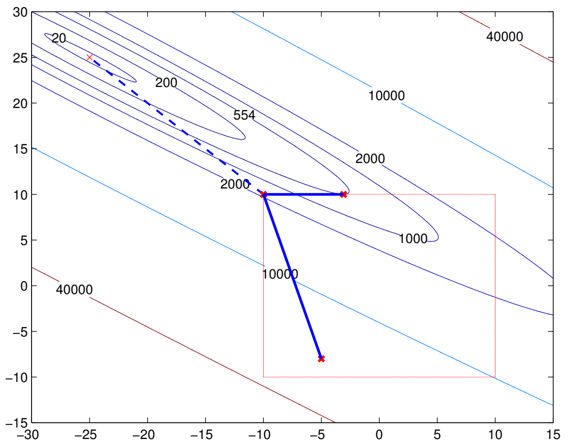
Fig. 3. The two-dimensional example (17), solved using the modified algorithm.

The problem was solved using Algorithm 1, implemented in Matlab code in the Quadratic Control Allocation Toolbox (QCAT) [13], as well as with Algorithm 2.

The operation of the standard algorithm is illustrated in Figure 2. From an initial starting point, the next iterate is found by following the line connecting the starting point and the unconstrained minimum until the boundary of the feasible set is encountered. At this point a new constraint, $u_1 = \underline{u}_1 = -10$ is added to the working set. In the next iteration, u_2 is the only free variable, and the minimum of the unconstrained cost function in this subproblem is also outside the feasible set, so the constraint $u_2 = \overline{u}_2 = 10$ is added to the working set. In the following iteration, there are no free variables, so the constraint with the most negative Lagrange multiplier ($u_1 = \underline{u}_1$) is removed from the working set. In the final iteration, the unconstrained minimizer is feasible. The KKT conditions are checked and found to be fulfilled, and the algorithm terminates.

Figure 3 illustrates the operation of the modified algorithm, starting from the same initial point. In the first iteration, the point in the feasible set closest to the unconstrained minimum is found. Although two constraints are active at this point ($u_1 = \underline{u}_1$ and $u_2 = \overline{u}_2$), only the constraint on u_2 fulfills the KKT conditions, and is added to the active set. In the following iteration the minimum is found to be feasible, and the KKT conditions are satisfied. This simple example illustrates the advantages of the modified algorithm over the standard algorithm, principally that fewer iterations are required.

B. Vehicle Dynamics Control

A more advanced example involving control allocation is a vehicle dynamics controller [7], [6]. Such control systems seek to stabilize roll and/or yaw dynamics using individual wheel braking. The control design may be carried out regarding the total forces and moments F_{xT}, F_{yT} and M_T as virtual controls. A control allocator is then used to map these signals onto braking commands for each wheel. The constraints are determined by the available friction, which in turn depends on the instantaneous normal force acting on each wheel. Rate constraints related to the hydraulic braking system may also be present. By considering Figure 4, the

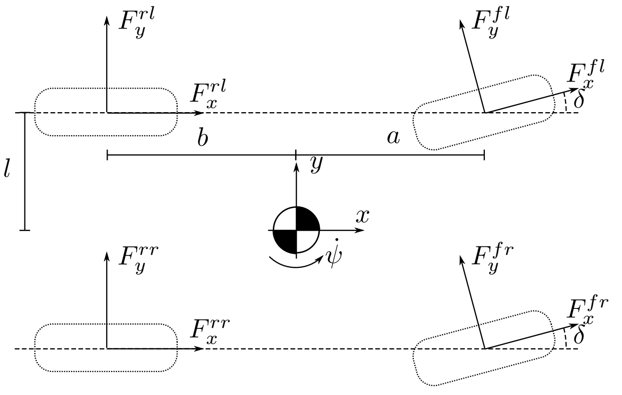

Fig. 4. Planar chassis model, showing the horizontal components of the tire forces.

following expressions relating the individual tire forces to the generalized forces are obtained:

$$F_{xT} = F_x^{rl} + F_x^{rr} + (F_x^{fl} + F_x^{fr})\cos\delta - (F_y^{fl} + F_y^{fr})\sin\delta \tag{18a}$$

$$F_{yT} = F_y^{rl} + F_y^{rr} + (F_y^{fl} + F_y^{fr})\cos\delta + (F_x^{fl} + F_x^{fr})\sin\delta \tag{18b}$$

$$\begin{aligned} M_T =& (F_y^{fl} + F_y^{fr})a\cos\delta + (F_x^{fl} + F_x^{fr})a\sin\delta \\ &- (F_y^{rl} + F_y^{rr})b + (F_x^{rr} + F_x^{fr}\cos\delta + F_y^{fl}\sin\delta \\ &- F_x^{rl} - F_x^{fl}\cos\delta - F_y^{fr}\sin\delta)l \end{aligned} \tag{18c}$$

where δ is the steering angle (measured at the wheels). The actual control variables are the longitudinal tire forces:

$$u = \begin{pmatrix} F_x^{fl} & F_x^{fr} & F_x^{rl} & F_x^{rr} \end{pmatrix}^T$$

The lateral forces F_{yT} are non-linearly related to the longitudinal forces, but a linear approximation can be used [7]. This gives a linear relationship between virtual and actual controls, $v = Bu$. The constraints are given by:

$$-|\sigma \mu F_{zi}| \leq u_i \leq 0 \tag{19}$$

where σ is a parameter of the tire characteristic approximation. This control allocation formulation was used in conjunction with a rollover mitigation controller described in [7]. A standard test maneuver used for investigating vehicle rollover was simulated using DaimlerChrysler's proprietary CASCaDE simulation software to test the controller. Both the standard active set algorithm and the modified version were used to solve the control allocation problem. Using hotstarting in this application leads to problems due to the time-varying constraints. This may be rectified by performing checks of feasibility and adjusting the starting point accordingly. However, there are many permutations of how this may be done, which will not be discussed further here. Figure 5 illustrates the distribution of the number of iterations required by the respective algorithms, with and without hotstarting. The modified algorithm most often requires fewer iterations than the standard algorithm, and never required more than six iterations, one less than the theoretical worst case of seven. The average number of iterations required by the standard algorithm without hotstarting was 4.9, while the average for the modified algorithm was 3.4. With hotstarting, and the additional logical checks described above, the average number of iterations required were 2.9 for the standard algorithm and 2.4 for the proposed algorithm.

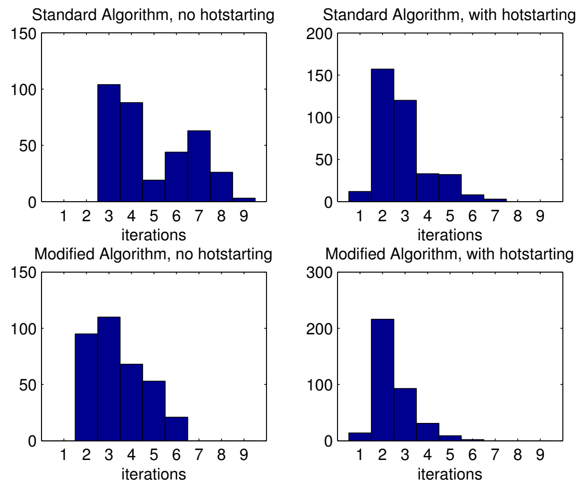

Fig. 5. Histogram showing the number of iterations required by the standard and modified algorithms during the VDC simulation.

C. Discussion

The results clearly indicate the improved performance of the proposed algorithm. Good performance is achieved without the need for hotstarting, which is important in this application where user checks are required to prevent problems with hotstarting. It can be seen that the particular hotstarting method used here improved performance, but in general it is desirable to avoid hotstarting in such applications.

VI. CONCLUSIONS

An active set algorithm for solving bound-constrained least squares problems has been presented. Unlike existing algorithms, the proposed algorithm does not require hotstarting to obtain good performance. This is particularly important in time-varying allocation problems, where the use of hotstarting can cause problems. The key feature in the proposed algorithm is the method for updating the iterates, which allows the addition of multiple constraints to the active set at each iteration. The algorithm has been tested on a real-world example, involving a time-varying control allocation problem in which other allocation methods, such as multi-parametric quadratic programming, encounter problems.

VII. ACKNOWLEDGMENTS

The work was carried out in the context of the Complex Embedded Automotive Control Systems (CEmACS) project, FP6-IST-004175. The author would also like to thank the reviewers for their valuable comments.

REFERENCES

[1] J. B. Davidson, F. J. Lallman, and W. T. Bundick, "Real-time adaptive control allocation applied to a high performance aircraft," in *Proceedings of the 5th SIAM Conference on Control & Its Applications*, 2001.

[2] J. Tjønnås and T. A. Johansen, "Optimizing nonlinear adaptive control allocation," in *IFAC World Congress*, Prague, Czech Republic, 2005.

[3] O. Härkegård, "Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation," in *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada USA, December 2002.

[4] J. H. Plumlee, D. M. Bevley, and A. S. Hodel, "Control of a ground vehicle using quadratic programming based control allocation techniques," in *Proceedings of the American Control Conference*, Boston, Massachusetts, USA, 2004.

[5] P. Tøndel and T. A. Johansen, "Control allocation for yaw stabilization in automotive vehicles using multiparametric nonlinear programming," in *Proceedings of the American Control Conference*, Portland, Oregon, USA, 2005.

[6] B. Schofield, T. Hägglund, and A. Rantzer, "Vehicle dynamics control and controller allocation for rollover prevention," in *Proceedings of the IEEE International Conference on Control Applications*, Munich, Germany, Oct. 2006.

[7] B. Schofield, "Vehicle dynamics control for rollover prevention," Department of Automatic Control, Lund University, Sweden, Licentiate Thesis ISRN LUTFD2/TFRT--3241--SE, Dec. 2006.

[8] K. A. Bordignon, "Constrained control allocation for systems with redundant control effectors," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 1996.

[9] O. Härkegård, "Backstepping and control allocation with applications to flight control," Ph.D. dissertation, Department of Electrical Engineering, Linkping University, SE–581 83 Linkping, Sweden, May 2003.

[10] Å. Björck, *Numerical methods for least squares problems*. SIAM, 1996.

[11] J. J. Moré and G. Toraldo, "Algorithms for bound constrained quadratic programming problems," *Numeriche Mathematik*, vol. 55, pp. 377–400, 1989.

[12] W. W. Hager, C. Shih, and E. O. Lundin, "Active set strategies and the lp dual active set algorithm," Department of Mathematics, University of Florida, Gainesville, FL, August 1996.

[13] O. Härkegård. (2004) Quadratic programming control allocation toolbox (qcat). [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/