Real-Time Optimization for use in a Control Allocation System to Recover from Pilot Induced Oscillations

Michael W. Leonard*

Science Applications International Corporation, Moffett Field, 94035, U.S.

Integration of the Control Allocation technique to recover from Pilot Induced Oscillations (CAPIO) System into the control system of a Short Takeoff and Landing Mobility Concept Vehicle simulation presents a challenge because the CAPIO formulation requires that constrained optimization problems be solved at the controller operating frequency. We present a solution that utilizes a modified version of the well-known L-BFGS-B solver. Despite the iterative nature of the solver, the method is seen to converge in real time with sufficient reliability to support three weeks of piloted runs at the NASA Ames Vertical Motion Simulator (VMS) facility. The results of the optimization are seen to be excellent in the vast majority of real-time frames. Deficiencies in the quality of the results in some frames are shown to be improvable with simple termination criteria adjustments, though more real-time optimization iterations would be required.

I. Introduction

Actuator rate saturation is an important issue for closed loop control systems since it may create significant phase shift and amplitude reduction between the commanded and actual system states. Phase shift manifests itself as an effective time delay which, in general, decreases the phase margin of the system. In flight control systems, effective time delay due to actuator rate saturation is considered a major instigator of Pilot Induced Oscillation (PIO) events. A PIO can be described as an inadvertent, sustained aircraft oscillation resulting from an abnormal joint enterprise between the aircraft and the pilot (see Ref. 1 for an introduction to the subject of PIO).

Phase compensation is the definitive measure of all recently proposed methods of PIO recovery. Most of these methods focus on the single-input single-output (SISO) application and are shown to be quite effective in this case (consider, for example, the Differentiate-Limit-Integrate (DLI) method described in Refs. 2–6). However, none of the SISO phase compensation methods has been effectively extended to multi-input multi-output (MIMO) systems, particularly those in which redundant actuators are available to realize the controller input.

The Control Allocation technique to recover from Pilot Induced Oscillations (CAPIO) System, however, is a method for phase compensation that is *designed* for MIMO systems with redundant actuators.[7–10] The CAPIO System achieves two aims simultaneously: 1) Phase compensation is applied to the controller input; and 2) The phase compensated total control effort is distributed between all of the available actuators. This approach is fundamentally different from an extended SISO approach where phase compensation and control allocation are applied in series. The CAPIO System, moreover, does not impede the utilization of redundant actuators to achieve secondary control system aims such as drag minimization or reconfiguration after a failure.

Utilization of the CAPIO System presents a challenge to the control system designer, however, because the CAPIO formulation requires that constrained optimization problems be solved at the controller operating frequency. In this article, we discuss how this challenge was met at the NASA Ames Vertical Motion Simulator

*Formerly Senior Simulation Engineer, Science Applications International Corporation (SAIC) at NASA Ames Research Center, MS 243-6, Moffett Field, CA 94035; currently Senior Member of Consulting Staff, Mentor Graphics, Inc., 46871 Bayside Parkway, Fremont, CA 94538, `michael_leonard@mentor.com`.

(VMS) facility where CAPIO was successfully integrated into a Short Takeoff and Landing Mobility Concept Vehicle being developed by industry under government sponsored studies.

The possibility of solving constrained optimization problems at the controller operating frequency, or in *real time* as part of a piloted simulation, stems in part from the special nature of the CAPIO optimization problem. The objective function q is quadratic with a symmetric positive definite Hessian (see Sect. II) subject only to simple bound constraints. Despite the fact that the bounds add greatly to the complexity of the much simpler unconstrained problem, very efficient methods have been derived to solve the bound-constrained quadratic program (BQP), as is discussed in Sect. III.

This article is organized as follows. Section II describes the CAPIO System in greater detail with emphasis on the formulation of BQP. Section III gives a brief survey of the methods that can be used to solve BQP, concluding with the applicability of L-BFGS-B. Section IV describes how L-BFGS-B is successfully integrated into a real-time simulation. This section includes an overview of L-BFGS-B (reference Sect. IV.A); a summary of the technical details of the integration (reference Sect. IV.B); a specification of economic q and ∇q formulations, which are used to provide the BQP function value and gradient to L-BFGS-B (reference Sect. IV.C); and a description of how L-BFGS-B is modified to more efficiently solve BQP (reference Sect. IV.D). Section V presents a summary of the performance of L-BFGS-B, an evaluation of the quality of the L-BFGS-B solutions, in particular. Detailed results from the piloted simulation in VMS are discussed in Ref. 28. A brief summary of the accomplishment including the real time impact of the CAPIO System and the applicability of this research to future experiments is given in Sect. VI.

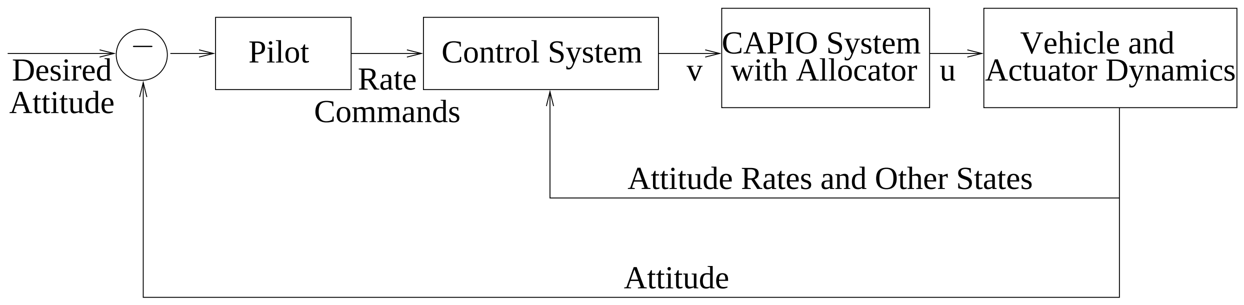
II. CAPIO System Details

Modern control systems require the efficient distribution of controller inputs to an increasingly larger number of redundant actuators, which in turn position control surfaces to realize the inputs. The resolution of actuator redundancy has been handled in two distinct but related ways: optimal control and control allocation (see Ref. 11 for complete details). The CAPIO System, or simply CAPIO, is an advanced formulation of the latter method. As such, it is considered to be distinct from the control system, which computes the controller input. CAPIO accomplishes two goals at the same time: allocation of the controller input to available actuators; and phase compensation in the case that actuator rate limiting is causing the controller input and the vehicle response to be out of phase. CAPIO consists of three parts: detection of phase lag, derivative-following engagement and disengagement, and the allocator. The allocator is to perform minimization of a quadratic objective function subject to bound constraints, a particular type of *quadratic program*. This section is to explain in some detail the formulation of the quadratic program.

The CAPIO allocator depends upon a mathematical relationship between the *controller input* and the *actuator command*, terms which are now briefly defined. The controller input is a 3-tuple of rotational accelerations that are related to control modes and/or pilot inputs. The actuator command can be thought of as a list of desired angles of the vehicle control surfaces such that the vehicle's angular accelerations due to the control surfaces are the controller input. (In actuality, there is a complicated nonlinear relationship between an actuator command and a control surface position. An actuator, driven by an electric or hydraulic motor, is used to position a control surface in accordance with a command from the allocator. Neglecting nonlinear particulars, an actuator position is analogous to a control surface position, but dynamics relate an actuator command to its achieved position, both as functions of time.) The CAPIO allocator derives from a state-space formulation of the vehicle and actuator dynamics (see Ref. 8 for a complete derivation). Certain aspects of the derivation are repeated now to provide a foundation for the discussion of our real-time implementation.

The notation used in the derivation of the CAPIO allocator are consistent with standard usage. We let m denote the control surface count, which is usually between 5 and 20 (see Bodson, Ref. [12, p. 2]). It is assumed herein that $m \geq 3$. Let $u(t) \in I\!R^m$ and $\delta(t) \in I\!R^m$ denote the commanded and actual actuator positions. The 3-tuple $v(t)$ is used to denote the total control effort, or vehicle rotation accelerations, demanded of the control surfaces by the control system. The role of the allocator is to determine the best way to use the available control surfaces in order to achieve the total control effort. Fig. 1 shows a simplified representation of the outer loop of a rate command attitude control system that employs an allocator to determine the desired positions of the actuators.

Continuing with the derivation, let n denote a number of vehicle states, and let $x(t) \in I\!R^n$ denote states other than the m actuator positions. The attitude rates $p(t)$, $q(t)$, and $r(t)$ are assumed to occupy the



Desired Attitude
−
Pilot
Rate Commands
Control System
v
CAPIO System with Allocator
u
Vehicle and Actuator Dynamics
Attitude Rates and Other States
Attitude

Figure 1. Attitude Control with Rate Commands

first three positions of $x(t)$. The relevant state transition matrices are $A(x) \in I\!R^{n\times n}$, $B_x(\delta) \in I\!R^{n\times m}$, and $B_\delta(\delta) \in I\!R^{m\times m}$. In terms of these quantities, a linearized state-space model of the vehicle and actuator dynamics is given by

$$\begin{array}{rcrcl} \dot{x}(t) & = & Ax(t) & + & B_x\delta(t) \\ \dot{\delta}(t) & = & & & B_\delta(\delta(t) - u(t)). \end{array} \tag{1}$$

The matrix B_x can be partitioned according to the three topmost rows that influence $\dot{p}(t)$, $\dot{q}(t)$, $\dot{r}(t)$, and the trailing rows as follows:

$$B_x = \begin{pmatrix} B \\ B_2 \end{pmatrix}, \qquad \text{where} \qquad B \in I\!R^{3\times m} \qquad \text{and} \qquad B_2 \in I\!R^{(n-3)\times m}$$

(there continues to hold the assumption that $m \geq 3$). In order to simplify the discussion, it is assumed that B has full row rank, though the method ultimately does not depend upon this.

Following Härkegård and Glad (see Ref. 11), two simplifications are made in order to derive a state-space formulation for use in control allocation. The first is to neglect actuator dynamics and the second is to consider the control surfaces to be pure moment generators. These simplifications apply only to the formulation of the allocator objective function; they would not be applied to the vehicle or actuator models. The effects of these simplifications are that $u(t)$ is equal to $\delta(t)$ and that $B_2 = 0$. It follows that Eq. (1) simplifies to

$$\dot{x}(t) = A\,x(t) + \begin{pmatrix} B\,u(t) \\ 0 \end{pmatrix}, \qquad \text{where} \qquad B\,u(t) \in I\!R^3.$$

The product $B\,u(t)$ constitutes the portions of the vehicle angular accelerations attributable to the control surfaces. As such, $B\,u(t)$ is exactly what the controller commands. That is, we have arrived at the simplified result that $u(t)$ should ideally be a solution to the system $B\,u(t) = v(t)$, where $v(t)$ is the aforementioned total control effort required by the controller. As B is assumed to have full row rank, such a solution exists.

Consistent with Bodson (see Ref. 12), we restrict our attention to a single real-time frame in the remainder of the derivation. To this end, let j denote the index of the current iteration of real-time simulation and t_0, t_1, …, t_{j-1}, t_j denote discrete values of time satisfying $t_j = t_{j-1} + \delta t$, where δt denotes the period of each frame. At time t_j, the allocator is required to provide a vector $u_j \in I\!R^m$, which is not a function of t, that depends upon a vector $v_j \in I\!R^3$, which is provided by the controller. Notationally, however, it is convenient to use u in place of u_j. The convention used in the remainder of this section is that variables without subscripts denote quantities at time t_j. Underlined variables without subscripts denote quantities at time t_{j-1}. For example, $\underline{u}$ is used to denote u_{j-1}.

The remainder of the derivation is drawn from Yildiz, Kolmanovsky, and Acosta (see, e.g., Ref. 8). At each frame, the allocator is required to solve $Bu = v$, or to approximate a solution depending upon additional considerations as follows:

- Smaller normed u are favored as these imply less control effort. This stipulation can be addressed directly by various techniques to find the minimum-norm solution of $Bu = v$, or a minimum-norm solution can be approximated using optimization applied to the objective function $q_c(u) = \|W_p^{1/2}(Bu - v)\|_2^2 + \epsilon\|u\|_2^2$, where $\epsilon > 0$, q_c is used to denote a "conventional" quadratic, and W_p is a nonnegative scaling matrix.

- Bounds on u directly related to position and rate limits of the actuators are imposed at every frame. Let b_l and b_u denote the lower and upper bounds, respectively. The requirement $b_l \le u \le b_u$ can put solutions of $Bu = v$ out of reach, but the objective function q_c introduced in the previous item is still applicable.

Phase shift is an important consideration that uniquely characterizes the CAPIO System allocator. Phase compensation is achieved through the novel inclusion of a derivative-following term in the objective function. As u and v are vectors of scalars, we introduce the notation u^p and v^p to denote notional differentiated (primed) quantities. In terms of these quantities, the following term is included in the objective function to be minimized by the CAPIO allocator:

CAPIO Allocator Phase Compensation Term

$$\|W_d^{1/2}(Bu^p - v^p)\|_2^2, \quad \text{where} \quad u^p = \frac{u - \underline{u}}{\delta t}, \quad v^p = \frac{v - \underline{v}}{\delta t}, \quad \text{and} \quad W_d \in I\!R^{3\times 3} \tag{2}$$

is a nonnegative scaling matrix.

With these considerations in mind, flight control with the CAPIO System uses the hybrid objective function given by

$$q(u) = \|W_p^{1/2}(Bu - v)\|_2^2 + \|W_d^{1/2}(Bu^p - v^p)\|_2^2 + \epsilon\|u\|_2^2 \tag{3}$$

(see Ref. 28). The scaling matrix W_p is normally fixed with positive diagonal elements. W_d is 0, unless the CAPIO System detects phase lag, in which case derivative-following engagement and disengagement is activated. As a diagonal element of W_d changes from zero to a positive value and returns to zero, the derivative following behavior of the CAPIO allocator is engaged and disengaged, respectively.[28]

The objective function q can be written in terms of a positive-definite Hessian as follows:

$$q(u) = \frac{1}{2}u^T H u + c^T u + d, \tag{4}$$

where H, c, and d are given by

$$\begin{aligned} H &= 2B^T(W_p + \frac{1}{\delta t^2}W_d)B + 2\epsilon I_{m\times m}, \\ c &= -2(B^T W_p v + \frac{1}{\delta t^2}B^T W_d B\underline{u} + \frac{1}{\delta t}B^T W_d v^p), \quad \text{and} \\ d &= \frac{1}{\delta t^2}\underline{u}^T B^T W_d B\underline{u} + \frac{2}{\delta t}\underline{u}^T B^T W_d v^p + \|W_p^{1/2}v\|_2^2 + \|W_d^{1/2}v^p\|_2^2. \end{aligned} \tag{5}$$

We are led finally to the CAPIO allocator problem BQP

Bound-constrained Quadratic Program (BQP)

$$\begin{aligned} \text{minimize} \quad & q(u) = \tfrac{1}{2}u^T H u + c^T u + d \\ \text{subject to} \quad & b_l \le u \le b_u, \end{aligned} \tag{6}$$

where $H \in I\!R^{n\times n}$ is positive definite.

III. Brief Survey of Methods to Solve BQP

The quadratic programming problem BQP given in Eq. (6) is a special case of subproblems that arise in sequential quadratic programming (SQP) methods[13] for the minimization of general nonlinear functions subject to nonlinear inequality constraints. The general quadratic programming problem, which may involve a semidefinite or indefinite Hessian as well as linear equality and inequality constraints, has been studied extensively and several solvers are available (see, for example, Ref. 14 and Ref. 15). Gill and Wong[16] discuss mathematical considerations in the determination of a point satisfying second-order necessary conditions for a general quadratic program. Quadratic programs constrained only by bounds are discussed in Ref. 17 and Ref. 18.

This paper proposes to use the limited-memory Broyden-Fletcher-Goldfarb-Shanno method extended to address bound constrained problems, known as L-BFGS-B, to solve BQP in real time (see Ref. 19 and Ref. 20 for the theoretical and implementation details of L-BFGS-B). An implementation of L-BFGS-B is easy to obtain, simple to use, and is proven to solve a great variety of bound-constrained optimization problems.[20]

L-BFGS-B is a compelling alternative for solving BQP, though minor modifications are applied for this real time application. First, the method's predecessor L-BFGS, which is intended for unconstrained optimization, can be guaranteed to converge in a finite number of iterations when applied to a quadratic of the form $q(u)$ (see Eq. (6)), a property known as quadratic termination. Second, L-BFGS-B is designed specifically for bound-constrained optimization. The focus on this particular type of optimization problem allows the authors of L-BFGS-B to incorporate efficiencies that might be inapplicable to an algorithm intended to solve problems with more general constraints. Third, an implementation of L-BFGS-B is readily available with no restrictions placed on its usage other than that the articles describing the algorithm and its implementation be properly referenced.

IV. Integration of L-BFGS-B into CAPIO

The Boeing Company has developed the Speed Agile Concept Demonstration (SACD) vehicle,[21] a specific version of the Short Takeoff and Landing Mobility Concept Vehicle being developed by industry under government sponsored studies. The SACD vehicle model is implemented primarily as a Simulink block diagram. The CAPIO System is implemented as a Simulink block diagram, which is integrated into the vehicle model with logic to enable switching between the CAPIO System and the native allocator. From this existing architecture came the primary requirement associated with the integration of L-BFGS-B into CAPIO: it must be executable from a Matlab Simulink block. It is of fundamental importance, in addition, that the block be able to be converted to real-time program code using Matlab's Real-Time Workshop, in order to support real-time piloted simulation in the Vertical Motion Simulator at the NASA Ames Research Center.

The process of integrating L-BFGS-B into CAPIO can be separated into three parts. The first part is to follow Matlab's Simulink Application Programmer's Interface (API) in order to support execution of the L-BFGS-B Fortran code base from a Simulink block diagram. The second part is to implement efficient Fortran code to provide L-BFGS-B with the objective function's gradient and function value, quantities which may need to be generated many times during a single execution frame. The third part is to modify L-BFGS-B in order to increase its efficiency when applied to problem BQP. In this section, a brief overview of L-BFGS-B is provided first. Then each of these three steps is elaborated upon as necessary to explain how the integration is achieved.

IV.A. Overview of L-BFGS-B

L-BFGS-B is a general-purpose solver for the problem

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && b_l \le x \le b_u, \end{aligned} \tag{7}$$

where $f : I\!R^n \to I\!R$ is a nonlinear function that is at least differentiable[19] (in this section, consistent with numerical optimization literature,[22] the variable $x \in I\!R^n$ denotes an array of real numbers, as distinguished from the states of a control system). The underlying algorithms synthesize and extend an extensive list of research articles on limited-memory quasi-Newton methods, and quasi-Newton methods in general. Quasi-Newton methods provide a robust middle ground between two extremes: the steepest descent method, which requires the provision of first derivatives only; and Newton's method, which requires the provision of both first and second derivatives. Quasi-Newton methods can be considered iterative secant methods for minimization–first derivatives only are required, and second-derivative information is approximated and accumulated as the iterations evolve. At each iterate x_k, line search quasi-Newton methods involve at least three procedures: the determination of a search direction p_k; an algorithm known as a line search to determine a step length α_k along p_k; and several updates, including the update $x_{k+1} = x_k + \alpha_k p_k$, in order to prepare for the next iteration.

Line search Quasi-Newton methods are carefully designed to generate descent directions. A descent direction p_k is such that a *positive* step length α_k (actually an interval from which the step length can be drawn) can be found such that $f(x_k + \alpha_k p_k)$ is strictly less than $f(x_k)$. For example, in the unconstrained case, a positive definite approximate Hessian H_k is maintained and updated, and p_k is the solution to the system $H_k\, p = -\nabla f(x_k)$, where $\nabla f(x_k) \neq 0$. The consequent condition $\nabla f(x_k)^T p_K < 0$ ensures the existence of the desired interval. The nonlinearity of f, coupled with the fact that only first derivative information is used in the definition of p_k, admits the possibility that $f(x_k + p_k)$ can be greater than $f(x_k)$. Many quasi-Newton methods are designed to overcome this difficulty by the inclusion of an iterative line search method to determine a positive step length α_k (potentially other than one) along p_k such that $f(x_k + \alpha_k p_k)$ is strictly less than $f(x_k)$ at least. Generally, it is advantageous both theoretically and practically to enforce stricter criteria on the step length (see Fletcher, Ref. [23, pp. 26–33], for a complete discussion). L-BFGS-B employs Moré and Thuente's well-known line search,[24] which is designed to enforce the Wolfe conditions (see Wolfe, Ref. 25) whenever possible. Theoretically, the imposition of these criteria at each step of a quasi-Newton method for unconstrained optimization is used to ensure global and superlinear convergence. In practice, L-BFGS-B is shown to be substantially more efficient when the Wolfe conditions are enforced as often as possible.[19] Both theoretical and practical considerations suggest that the line search be initialized with 1 (see Ref. [22, p. 128]).

IV.B. Use of Matlab's Simulink API to execute L-BFGS-B

The CAPIO System is implemented as a Simulink block diagram in order to support its integration into the flight control system of Boeing's Speed Agile Concept Demonstration (SACD) vehicle, which is implemented as a fully integrated Matlab/Simulink simulation environment.[21] This requires that the L-BFGS-B Fortran code base be accessible from a Simulink block. Matlab supports this kind of code accessibility through the use of the Level-2 Matlab S-Function Application Programmer Interface (API). This API is comprised of a Matlab function and a Level-2 Matlab S-Function Simulink block. The Matlab function itself comprises a set of callback methods that the Simulink engine invokes when updating or simulating the CAPIO System model. These callbacks define the inputs and outputs for the corresponding Simulink block. Once the callbacks have been implemented, the Matlab function is realized as a compiled object, which is executed as needed by any Simulink model that contains the corresponding Simulink block. Figure 2 illustrates a conceptual view of the Matlab S-Function Simulink Block as it would appear within a Simulink model. In this view of the Simulink block, most of the inputs and outputs have already been defined (see Eqs. (5)). Stars are used to denote optimal quantities, as determined by L-BFGS-B. The text variables, such as `iprint`, correspond to variables in the L-BFGS-B code base,[20] except for `capioOn`, which denotes an input to trigger real-time optimization, or disable it in favor of a standard allocator. The output $\|\mathrm{proj}\, \nabla q^*\|_\infty$ supplies the infinity norm of the projected gradient at the final iterate, for use in diagnosing the quality of each solution (see Ref. 20 for more information about this quantity).

The allocator does not require the implementation of any states or derivatives (using Simulink's callback terminology) because its outputs are a function of its inputs at each frame, as defined by Eqs. (5). For this reason, the only callback required to complete the implementation of the Level-2 Matlab S-Function is one to provide the outputs as a function of the inputs. Simulink provides a tool called an S-Function Builder to facilitate development of the block shown in Fig. 2 as well as the required callback.

The callback to implement the outputs of the block shown in Fig. 5 executes the L-BFGS-B code base. This is accomplished by a C function call to a Fortran subroutine, which is modeled after one of the drivers supplied with L-BFGS-B (see Ref. [20, p. 3]). This Fortran subroutine allocates the memory needed by L-BFGS-B and implements a loop to execute the following functionality:

1. Call L-BFGS-B's highest-level subroutine, which is named `setulb`.
2. Interpret an L-BFGS-B message, `task`, which indicates primarily whether a line search has been completed (the first five elements of `task` would be `NEW_X` in this case)[a] or a line search is in progress (the first two element of `task` would be `FG` in this case). Otherwise, `task` may indicate that one of the

[a]In case L-BFGS-B has reached a new iterate, the opportunity to terminate the optimization based upon a preset limit on

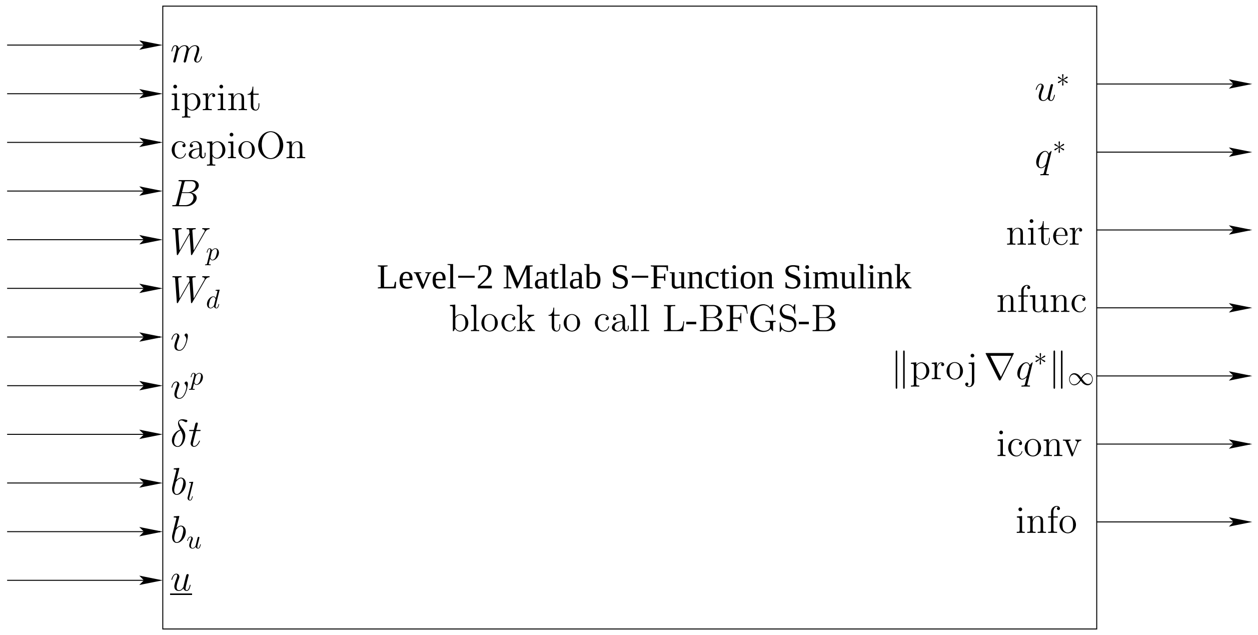


Figure 2. Level-2 Matlab S-Function Simulink Block to Solve BQP

termination criteria has been met or that an error condition has arisen. In either of these cases, the loop is terminated.

Whenever the first two elements of `task` are `FG`, the function value and gradient must be supplied to L-BFGS-B. This is accomplished by a call, from within the loop, to another Fortran subroutine. This latter subroutine is used to evaluate $q(x)$, as it is defined in Eq. (6) and Eqs. (5), and $\nabla q(x)$, where x is used to in this context to denote an iterate of the optimization. The next section describes how these values are efficiently supplied.

IV.C. Efficient implementation of q and ∇q

L-BFGS-B, and line search quasi-Newton methods in general, require a function value and gradient at each iteration of the line search. In terms of the optimization variable x, L-BFGS-B requires $q(x)$ and $\nabla q(x)$ in order to solve BQP. The BQP objective, given in Eq. (6), is repeated here, along with its gradient. These are

$$q(x) = \frac{1}{2}x^T Hx + c^T x + d \quad \text{and} \quad \nabla q(x) = Hx + c \tag{8}$$

where H, c, and d are given in Eqs. (5). It is important to notice that H itself is not needed by L-BFGS-B; only quantities involving Hx are required. Formulation of H requires approximately $3(m^2 + m)$ multiplications and additions, a pair of which is typically referred to as an *operation* for simplicity. Once H has been formed, $q(x)$ and $\nabla q(x)$ require approximately $m^2 + m$ additional operations. Alternatively, formulation of Hx using B, W_p, and W_d takes approximately $6m$ operations (form Bx using $3m$ operations, followed by $(W_p + (1/\delta t^2)W_d)$ times Bx, followed by premultiplication by B^T using another $3m$ operations). (The quantity c is formed whether or not H is formed, at a cost of approximately $6m$ operations.) Table 1 provides a comparison between the two approaches.

Matlab's Optimization Toolbox includes a method known as `quadprog`, which can be used to solve BQP in the context of a CAPIO System implemented as a Simulink model. The arguments of `quadprog` include H, which means that it must be formed explicitly. The method `quadprog` is not supported by Matlab's Real-Time Workshop, however, so any real time implementation of the CAPIO allocator must use an alternative

the number of iterations or function evaluations presents itself. This method of termination, however, was not used by the CAPIO allocator.

Table 1. Operations to evaluate $q(x)$ and $\nabla q(x)$ using H versus using B

Computation	Using H	Using B
Forming H	$3(m^2+m)$	N/A
Evaluating Hx	m^2+m	$6m$
Total	$4(m^2+m)$	$6m$

method. In an early implementation of the CAPIO System, built for desktop fast-time use, `quadprog` had been used. Direct substitution of a Level-2 S-Function for `quadprog` was simplified by the continued use of H as an input to the S-Function. Because m is significantly larger than 3, however, the numbers of calculations required at each frame of the simulation to support the allocator were prohibitive in real time. Using B directly in order to decrease the required numbers of operations per frame was a first step towards realizing a real-time implementation of the CAPIO allocator. In the next section, a second step, involving modifications to L-BFGS-B itself, is presented.

IV.D. Modifications of L-BFGS-B to efficiently solve BQP

Problem BQP involves a Hessian that is positive definite, which implies that a very specialized *exact line search* can be applied, in place of the standard line search included with L-BFGS-B. It has already been described that standard line search functionality is to determine α_k such that $f(x_k+\alpha_k p_k)$ is less than $f(x_k)$ and, more strictly, such that the Wolfe conditions are satisfied (reference Sect. IV.A). Consider here the univariate function

$$l(\alpha) = q(x_k + \alpha p_k), \tag{9}$$

where q is given by Eq. (6) and p_k is a search direction generated by L-BFGS-B. The derivatives of this function are given by

$$l'(\alpha) = \nabla q(x_k)^T p_k + \alpha p_k^T \nabla^2 q(x_k) p_k \quad \text{and} \quad l''(\alpha) = p_k^T \nabla^2 q(x_k) p_k. \tag{10}$$

The matrix $\nabla^2 q(x_k)$ is the constant H. Because H is positive definite and p_k is nonzero (L-BFGS-B terminates with an error condition otherwise), $l''(\alpha) > 0$, which means $l(\alpha)$ has a unique minimizer. This minimizer, denoted herein by α^*, and found by solving $l'(\alpha) = 0$, is given by

$$\alpha^* = -\frac{\nabla q(x_k)^T p_k}{p_k^T H p_k}, \tag{11}$$

which is positive since p_k is a descent direction (reference Sect. IV.A).

In general, even when f is convex, the approximation of a positive minimizer of $f(x_k + \alpha\, p_k)$ can be costly. Moreover, both theoretical and practical analyses indicate this this is not a desirable strategy (see Ref. [22, p. 126]). In this case, however, α^* can be evaluated directly at a potentially acceptable cost, depending upon whether or not its use decreases the profile of L-BFGS-B overall. In the special case when a quasi-Newton method is used to minimize a quadratic with positive definite Hessian (such as $q(x)$, where q is given by Eq. (6)), an exact line search can lead to the method's termination at the minimizer in a finite number of steps, about m in this case[b]. The special nature of q, therefore, suggests that modifying the L-BFGS-B line search to utilize α^* as its first iterate instead of the unit step may be beneficial. The value α^* may not be admissible, however, because it could imply a bound constraint violation. The same holds for the unit step length, of course, and the same remedy applies: initialize the line search with the minimum of α^* and the step to the nearest bound constraint along p_k. Though it can be shown that α^* satisfies the Wolfe conditions, this result is not used in the suggested implementation. Rather, the only modified aspect of L-BFGS-B is to use α^* in place of 1 as the initial choice of step length, at the same point that the unit step is selected normally (just before the point where L-BFGS-B checks the initial choice of step length against the step to a nearest bound).

[b]This property, which applies in particular to a quasi-Newton method that uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update (see Ref. [22, p. 119]), is known as *quadratic termination.*

The evaluation of α^* requires a second matrix-vector multiplication at each iteration of L-BFGS-B, this to obtain Hp_k. As before, it is more economical to use B instead of H (see Table 1), and when B is used the cost is approximately $6m$ operations. The implementation used to evaluate $q(x)$ and $\nabla q(x)$ is immediately applicable as long as L-BFGS-B is modified to query the user's driver for $p_k^T H p_k$ in a manner similar to the way it asks for $q(x)$ and $\nabla q(x)$. In the latter case, L-BFGS-B puts the first two elements of `task` to `FG` (see Sect. IV.B). In order to evaluate α^*, L-BFGS-B was modified to put the first four elements of `task` to `DTHD` at the start of a line search, return to the driver to use the applicable portion of the implementation providing $q(x)$ to obtain $p_k^T H p_k$ (by putting x to p_k and evaluating $x^T H x$), reenter L-BFGS-B at the point where the initial step is normally set to 1, finish the evaluation of α^*, and proceed from there as usual.

With these modifications, wherein L-BFGS-B performs an exact line search at each iteration, L-BFGS-B is seen to use significantly fewer iterations and function evaluations when applied to BQP problems with randomly generated parameters. When these changes are applied, and the economical evaluations of $q(x)$ and $\nabla q(x)$ described in Sect. IV.C are implemented, the resulting real-time implementation of the CAPIO System within the SACD simulation environment was able to executed within the VMS real-time framework in order to support piloted evaluations. The next section is to detail how certain L-BFGS-B parameters were chosen for use by the real-time allocator and to describe how well L-BFGS-B performed during a series of piloted check-out runs.

V. Performance of the CAPIO Allocator

Methods for numerical optimization are generally iterative algorithms that terminate when certain stopping criteria are met. (See Gill, Murray, Wright [22, pp. 305–312] for a detailed discussion of termination criteria.) At least one of the criteria of typical quasi-Newton methods is related to the first-order necessary optimality criteria. For example, a method for unconstrained optimization may be designed to terminate at an iterate x_k when $\|\nabla f(x_k)\| <$ `tol`, where `tol` is a small, positive number. This criterion reflects the fact that if x^* is a local minimizer of $f(x)$, then $\|\nabla f(x^*)\|$ is zero[c]. But the criterion also reflects the fact that the iterative nature of the quasi-Newton method, and the presence of round-off error in all evaluations of $f(x)$ and $\nabla f(x)$, prevent the realization in general of an iterate x_k where $\nabla f(x_k)$ is exactly equal to zero.

V.A. Development of termination criteria

L-BFGS-B provides two built-in stopping criteria[20] as follows:

$$\|\mathrm{proj}\,\nabla f(x_k)\|_\infty \le \texttt{pgtol}, \tag{12}$$

where `pgtol`, which stands for *projected gradient tolerance*, is set by the user, and

$$\frac{f(x_k) - f(x_{k+1})}{\max(|f(x_{k+1})|, |f(x_k), 1)} \le \texttt{factr} * \texttt{epsmch}, \tag{13}$$

where `epsmch` is the machine precision and `factr` also is set by the user. The first of these two criteria is associated with the first-order optimality criteria of the problem given in Eq. (7). The second is designed to stop the optimization when the change in f becomes sufficiently small, an indication that the sequence $\{x_k\}$ may be converging. Typical values for `pgtol` and `factr` are 10^{-8} and 10^7, respectively (a smaller value of `pgtol` or `factr` in Eq. (12) or Eq. (13), respectively, implies a more stringent criterion in either case).

Another related criterion[22] reflects the difficulty of driving the norm of the gradient to a small value when the objective function is large at a minimizer:

$$\frac{\|\mathrm{proj}\,\nabla f(x_k)\|_\infty}{\max(|f(x_k)|, 1)} \le \texttt{pgftol}, \tag{14}$$

where `pgftol` is set by the user. This criterion was not used during any of the piloted VMS runs, but it is used in the following discussion of the allocator's performance.

L-BFGS-B terminates if either one of the criteria given in Eqs. (12) and (13) is met. Determination of suitable values for `pgtol` and `factr` is at the discretion of the user of L-BFGS-B. Putting either one of these values to zero disables the associated criterion. Putting both to zero would allow the user to

[c]The objective f is assumed to be sufficiently smooth in an open convex set in $I\!R^n$ that contains x^*.

choose other criteria and/or to implement more complicated logic involving two or more criteria. For the purpose of solving BQP, the default criteria and logic of L-BFGS-B were left in place. A standalone Matlab Simulink simulation containing the CAPIO allocator, capable of solving randomly-generated BQPs over any number of frames, was used to determine values of `pgtol` and `factr` for use during the piloted VMS runs. Equation (12) was considered the preferred criterion and it was found that the allocator could terminate often with `pgtol` as small as 10^{-10} in a tolerable number of frames. (The tolerable number of frames had been determined to be around 20 in a complete real-time SACD simulation utilizing the CAPIO System.) Some of the randomly-generated BQP problems were such that the allocator could not drive the projected gradient norm down to 10^{-10} quickly enough. But in those cases it was seen that the criterion in Eq. (13) could be met with `factr` as low as 10^6. Therefore, the CAPIO allocator was implemented with both of the criteria simultaneously in play with `pgtol`$= 10^{-10}$ and `factr`$=10^6$, respectively.

Practically speaking, it is important to have upper limits on the allowable numbers of iterations and function evaluations. These limits were implemented in the following manner. If L-BFGS-B returns to the driver prepared for a new iteration, as indicated when the first 5 values of `task` are `NEW_X`, the driver ends the optimization if both the number of iterations and the number of function evaluations are greater than two, respective, preset upper limits. These limits were set high enough (each to 100), however, that this criterion for termination was not achievable in real time. That is, it was decided that termination of the real-time simulation was to be preferred over ending the optimization prematurely. In practice, the optimization always terminated within the frame time limitations. In this sense, the limits on iterations and functions evaluations were out of reach.

V.B. Quality of Solutions

In order to prepare for a piloted evaluation of the CAPIO System in the VMS, a series of piloted checkout runs was accomplished beforehand. The termination criteria given in Eqs. (12) and (13) of Sect. V.A were used for the entire checkout series. Assessment of the CAPIO System utilized a matrix of configurations that included runs with the CAPIO allocator and other runs using SACD's native allocator instead. A total of 19 CAPIO System runs were completed, and this total included 200,230 frames of optimization. The maximum number of function evaluations required over all of the frames was 17, which was acceptable with respect to the frame time limitations.

In every frame, the optimization terminated with either Eq. (12) or Eq. (13) in effect, or occasionally with an abnormal line search termination, which was associated always with the indication `info`$= -9$.[20] The diagnostic `info`$= -9$ indicates that the line search did not recognize α^* as an acceptable step. More precisely, the line search did not terminate in a small number (3 instead of the default value of 20) of iterations, though α^* was used as its initial iterate. Table 2 provides codes to use herein for each of these three reasons for termination, as well as a code associated with Eq. (14). The code AbT is to denote "abnormal termination"

Table 2. Codes to indicate reasons for termination

Criterion	Eq. (12)	Eq. (13)	`info`$= -9$	Eq. (14)
Code	C0	C1	AbT	C0R

and the "R" in the code C0R means "relative," because the criterion involves the projected gradient norm relative to the absolute value of the objective function. A number of abbreviations are introduced in Table 3 to simplify the presentation.

Table 3. Abbreviations to simplify presentation

Expression	$f(x_k)$	$\nabla f(x_k)$	Last $f(x_k)$	Last $\nabla f(x_k)$	$\|\cdot\|_\infty$
Abbreviation	f_k	g_k	f_t	g_t	$\|\cdot\|$

The first five columns of Table 4 provide a summary, for each CAPIO allocator checkout run, of the numbers of frames at which the optimization terminates with C0, C1, or AbT. The quality of a solution depends on which of the termination criteria triggers termination of the optimization. Any CAPIO allocator problem that terminates based upon the criterion C0 is considered to have been successfully solved. Termination for the other two reasons merits further investigation. The condition AbT is considered first. In this case, it

Table 4. CAPIO Allocator Termination Conditions During Checkout Runs

		Termination reason			$\max\{\|g_t\|\}$		$\max\{\|g_t\|/(\lvert f_t\rvert+1)\}$		$\mathrm{cond}(H) > 10^8$	
Run	Frames	C0	C1	AbT	C1	AbT	C1	AbT	C1	AbT
1	5576	4208	1356	12	7e-01	3e-03	7e-01	3e-03	1345	12
3	6439	5082	1248	109	1e+00	3e-04	1e+00	3e-04	1248	109
5	5625	4122	1498	5	7e-01	2e-03	7e-01	2e-03	1488	5
6	12168	12146	22	0	1e-07	0e+00	1e-07	0e+00	0	0
7	13188	13188	0	0	0e+00	0e+00	0e+00	0e+00	0	0
10	8978	8492	486	0	1e-05	0e+00	1e-05	0e+00	0	0
11	9787	9555	232	0	1e-05	0e+00	1e-05	0e+00	0	0
12	13518	8903	4522	93	3e-02	7e-04	3e-02	7e-04	3912	93
13	16666	7344	9301	21	8e-01	7e-04	8e-01	7e-04	9076	21
14	8633	4905	3703	25	3e-01	1e-03	3e-01	1e-03	3546	25
16	9912	7498	2395	19	7e-01	1e-03	7e-01	1e-03	2239	19
18	19794	16147	3647	0	5e-03	0e+00	5e-03	0e+00	1622	0
19	18688	15445	3225	18	2e-01	6e-05	2e-01	6e-05	1802	18
23	8615	5646	2951	18	5e-01	3e-03	5e-01	3e-03	2684	18
25	8713	7277	1427	9	1e+00	9e-05	1e+00	9e-05	1209	9
26	6469	5139	1329	1	5e-01	8e-07	5e-01	8e-07	1129	1
27	3053	2366	686	1	5e-03	2e-05	5e-03	2e-05	622	1
28	10846	230	10562	54	4e-01	4e-04	4e-01	4e-04	10562	54
6b	13562	13534	28	0	3e-07	0e+00	3e-07	0e+00	0	0

is found that the final norms of the gradients are as large as 3×10^{-3}. This is illustrated in the seventh column of Table 4[d]. However, the final norms of the gradients *relative to the associated absolute values of objective function* are no larger that 7×10^{-7}, as is shown in the ninth column of Table 4. It follows that condition C0R is satisfied with `pgftol`$= 10^{-6}$ at the end of all checkout runs that terminate according to condition AbT. The condition C1 is considered next. In this case, it is found that the final norms of the gradients are as large as 1, though again an improved assessment is realized with the criterion C0R. It is of interest to know what triggers these two types of terminations, instead of the preferred type associated with Eq.(12). Differentiating between the Hessians of the associated BQPs provides a compelling rationale: It is more difficult to meet criterion C0 when L-BFGS-B is applied to BQPs with ill-conditioned Hessians. This statement is effectively demonstrated by the last two columns of Table 4. In these columns, the numbers of frames where the condition number of H is greater that 10^8 are tabulated whenever the allocator terminates according to conditions C1 and AbT, respectively. Considering Run 28, for example, it is seen that the Hessian is ill-conditioned at *every* frame the allocator terminates according to condition C1 or AbT.

It is well known that quasi-Newton methods can require a disproportionately large numbers of iterations and function evaluations to converge to a minimizer where the Hessian is ill-conditioned (see, for example, Powell, Ref. 26 or Siegel, Ref. 27). In practice, this phenomenon is manifested by difficulty decreasing the gradient norm and by many successive iterates that differ little from each other, though they may be significantly far from a minimizer. The condition C1 can be the reason for termination when H is ill-conditioned because of this tendency of the method to stall. It is seen that with `factr` set to 10^6, the CAPIO allocator terminates with C0 or C1 almost always. The question of how the CAPIO allocator performs with a smaller value of `factr` arises naturally, however, and this is now addressed.

In order to determine the effect of setting `factr` to 1, the smallest value suggested by the authors of L-BFGS-B, a Matlab Simulink model containing a CAPIO allocator block was developed such that all of the BQP problems encountered during the checkout runs could be *solved again.* All of the information associated with the checkout run BQPs had been logged in real time, so this became a matter of populating the inputs

[d]The heading $\max\{\|g_t\|\}$ is shorthand for the maximum terminal gradient norm over all frames of a given run, and the subheading AbT, for example, is to denote this maximum over all frames where termination is as a result of condition AbT.

of the CAPIO allocator block (reference Fig. 2) with logged values and executing the block. This was done for every CAPIO run, and for every frame of each associated run. Table 5 illustrates the results of this "what-if" study. Several observations can be made:

Table 5. CAPIO Allocator Termination Conditions with `factr` set to 1

		Termination reason			$\max\{\|g_t\|\}$		$\max\{\|g_t\|/(\|f_t\|+1)\}$		$\text{cond}(H) > 10^8$	
Run	Frames	C0	C1	AbT	C1	AbT	C1	AbT	C1	AbT
1	5576	4535	923	118	1e-02	3e-03	1e-07	2e-07	923	118
3	6439	5422	806	211	3e-03	3e-04	7e-07	2e-08	806	211
5	5625	4517	997	111	2e-02	5e-03	3e-06	3e-08	996	111
6	12168	12168	0	0	0e+00	0e+00	0e+00	0e+00	0	0
7	13188	13188	0	0	0e+00	0e+00	0e+00	0e+00	0	0
10	8978	8978	0	0	0e+00	0e+00	0e+00	0e+00	0	0
11	9787	9787	0	0	0e+00	0e+00	0e+00	0e+00	0	0
12	13518	10735	2160	623	4e-03	2e-03	1e-06	7e-08	2160	623
13	16666	10601	5024	1041	2e-02	9e-03	4e-07	5e-07	5024	1041
14	8633	6606	1556	471	2e-02	4e-03	4e-07	5e-07	1554	471
16	9912	8872	738	302	2e-02	3e-03	1e-07	7e-07	737	302
18	19794	18296	1324	174	2e-03	2e-04	5e-07	8e-08	1324	174
19	18688	17465	973	250	4e-03	2e-03	1e-06	2e-07	961	250
23	8615	7396	1025	194	2e-02	3e-03	2e-07	1e-07	1025	194
25	8713	8172	472	69	2e-02	1e-03	3e-07	6e-09	472	69
26	6469	6006	403	60	1e-02	2e-03	2e-06	2e-08	403	60
27	3053	2812	216	25	2e-03	9e-04	2e-08	3e-09	216	25
28	10846	2528	7147	1171	3e-02	3e-03	1e-07	3e-08	7147	1171
6b	13562	13562	0	0	0e+00	0e+00	0e+00	0e+00	0	0

1. No new termination conditions are introduced.
2. Some iterations that previously terminated according to condition C1 now terminate according to either condition C0 or AbT. The majority of changes were from C1 to C0.
3. Problems where the allocator continues to terminate with condition C1 are associated with smaller final gradient norms and smaller relative gradient norms.

These observations suggest that the use of `factr`$= 1$ improves the quality of the solutions. First, the maximum value of the gradient norms associated with frames at which termination is due to condition C1 decreases from 1 to 2×10^{-2}. This is a statement regarding the frames where criterion 1 remains in effect. Second, during many more frames than before, criterion C0 is realized instead of criterion C1. Third, although many more frames are terminated with criterion AbT in effect, the associated final relative gradient norm maximum is decreased from 3×10^{-3} to 7×10^{-7} (compare the ninth columns of Tables 4 and 5).

The final iterate of a method that terminates according to criterion C1 can be especially sensitive to the choice of `factr` when the Hessian of the objective function is ill-conditioned at a minimizer (see [22, p. 306]). Since most of the BQPs that are terminated according to criterion C1 involve ill-conditioned Hessians, it is of interest to understand how a change in `factr` from 10^6 to 1 may affect the associated solutions. The fourth column of Table 6 gives the maximum relative change in the final iterates associated with all frames where BQP terminates according to criterion C1 when `factr` is equal to 10^6. In Table 6, an overbar is used to denote a quantity associated with a BQP solution obtained with `factr` set to 1. The notation {C1} is shorthand for the subset of frame indices at which L-BFGS-B terminated according to condition C1 (with `factr`$= 10^6$) *during the checkout series*. For each checkout run, the set {C1} is a superset of the set of frame indices at which L-BFGS-B terminates according to condition C1 with `factr`$= 1$ at the desktop. But, since our interest is in how a solution of a frame's BQP would have changed with `factr` set to 1, regardless of the associated termination criterion, each subset {C1} is defined relative to a checkout run. The notation

Table 6. Changes in solutions and function evaluations at all frames that L-BFGS-B terminates with C1 (with `factr`$= 10^6$) when `factr` is decreased to 1

Run	Frames	C1	$\max\limits_{\{C1\}} \left\{ \dfrac{\|x_t - \bar{x}_t\|}{1 + \|x_t\|} \right\}$	$\max\limits_{\{C1\}} \left\{ \dfrac{\lvert f_t - \bar{f}_t \rvert}{1 + \lvert f_t \rvert} \right\}$	$\max\limits_{\{C1\}}\{n_f\}$	$\max\limits_{\{C1\}}\{\bar{n}_f\}$
1	5576	1356	5%	6.22e-05	12	31
3	6439	1248	100%	6.35e-06	12	14
5	5625	1498	5%	8.99e-07	14	31
6	12168	22	0%	3.20e-10	11	12
7	13188	0	0%	0.00e+00	0	0
10	8978	486	10%	8.25e-07	5	6
11	9787	232	6%	6.06e-07	8	9
12	13518	4522	21%	8.63e-07	9	13
13	16666	9301	33%	2.04e-05	13	28
14	8633	3703	7%	2.74e-06	11	20
16	9912	2395	7%	1.00e-04	12	19
18	19794	3647	21%	8.66e-07	10	15
19	18688	3225	14%	8.94e-07	8	14
23	8615	2951	16%	2.60e-05	13	30
25	8713	1427	25%	8.84e-06	9	23
26	6469	1329	30%	1.45e-06	12	30
27	3053	686	62%	3.38e-07	10	18
28	10846	10562	40%	2.16e-06	11	24
6b	13562	28	0%	3.25e-10	11	12

n_f denotes the number of function evaluations, roughly proportional to the CAPIO allocator's frame time requirement, and

$$\max_{\{C1\}}\{n_f\}$$

is the maximum n_f required over all frames where the optimization terminated according to criterion C1 during the checkout series.

The following observations follow from the Table 6 data:

1. The solution of a problem where the allocator terminates with condition C1 when `factr`$= 10^6$ may be significantly different when `factr` is put to 1, though the final value of the objective function differs by very little.

2. The solution of a problem where the allocator terminates with condition C1 may require more function evaluations when `factr` is equal to 1 versus when `factr`$= 10^6$.

These results supplement the conclusion that a smaller value of `factr` may result in *improved quality* of a solution with the further conclusion that it can *significantly alter* the solution, when H is ill-conditioned. Termination with a decreased value of `factr` is seen, however, to require many more function evaluations, and the increased CPU time required each frame to achieve this may have been prohibitive in the real-time framework of our experiment.

VI. Results and Conclusions

A motion-based, piloted simulation evaluation was completed to assess the impact of the CAPIO allocator on the pilot-aircraft-control system characteristics. The simulation evaluation represents a milestone following a two-year effort by NASA and NASA contractors to mature the CAPIO system from Technology Readiness Level (TRL) 1 to TRL 5. A total of 647 piloted runs, 318 of which utilized the CAPIO System, was accomplished in a 3-week period. These 318 runs encompassed a total of 2,946,644 frames, each of which required the solution of problem BQP using the CAPIO allocator, implemented in terms of L-BFGS-B. The maximum numbers of iterations and function evaluations required over all of these frames were 15 and 21, respectively, with the termination criteria set as described in Sect. V.A. The largest terminal value of the relative projected gradient norm over all of these frames was 1.52×10^{-3}. Only one frame-time overrun occurred. This was the last frame of experiment run 549.

The CAPIO system is intended to address problems, such as phase lag and consequent PIO potential, that arise from stringent actuator rate limits for multi-input, multi-output applications. The system does this by actively detecting and eliminating phase lag introduced by control surface rate limiting. Results from the simulation evaluation confirm that the CAPIO system successfully contributed to a reduction in the severity and duration of PIOs, and to an improvement in the pilots' perception of PIO tendencies and in the pilots' PIO ratings, as compared to the baseline allocator.[28]

As noted in 28, further maturation of the CAPIO system is needed to enable lowering the actuator rate limit requirements. Analytical studies are needed to enhance and guarantee the closed-loop stability properties of an aircraft, and to extend the CAPIO system to influence and improve overall system characteristics. The system integration and impact of the CAPIO system will also need to be demonstrated through piloted flight studies. If successful, the CAPIO system may allow aerodynamically efficient airframe designs that are currently unattainable due to technologically prohibitive control power requirements.

Acknowledgments

Funding for this research was provided by two sources within the NASA Aeronautics Research Mission Directorate: the Fundamental Aeronautics Program's Subsonic Fixed Wing Project and the Integrated Systems Research Program's Environmentally Responsible Aviation Project.

The author would like to gratefully acknowledge Diana M. Acosta and Steven D. Beard of the NASA Ames Research Center for supporting this work. Thanks to them, as well as Robert W. Craun of Mission Critical Technologies, Inc., for many illuminating discussions regarding the technical aspects of this research. And the author would like to extend thankful acknowledgments to Jorge Nocedal of Northwestern University and Marc Bodson of the University of Utah for their helpful email messages.

References

[1]McRuer, D., "Human dynamics and pilot-induced oscillations," Minta Martin Lecture, Massachusetts Institute of Technology, Cambridge, MA.

[2]Deppe, P., Chalk, C., and Shafer, M., "Flight evaluation of an aircraft with side and centerstick controllers and rate-limited ailerons," Tech. rep., Advanced Technology Center, Calspan Corp., Buffalo, NY, April 1994, Final Rept. 8091-2.

[3]Snell, S. A. and Hess, R. A., "Flight control system design and rate saturating actuators," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 90–96.

[4]Snell, S. A. and Hess, R. A., "Robust decoupled, flight control design with rate-saturating actuators," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 3, 1998, pp. 361–367.

[5]Chapa, M., *A nonlinear pre-filter to prevent departure and/or pilot-induced oscillations (PIO) due to actuator rate limiting*, Master's thesis, Graduate School of Engineering, Air Force Inst. of Technology (AU), Wright-Patterson AFB, OH, March 1999, No. AFIT/GAE/ENY/99M-01.

[6]Liebst, B. S., Chapa, M. J., and Leggett, D. B., "Nonlinear prefilter to prevent pilot-induced oscillations due to actuator rate limiting," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 740–747.

[7]Yildiz, Y. and Kolmanovsky, I. V., "A control allocation technique to recover from pilot-induced oscillations CAPIO due to actuator rate limiting," *2010 American Control Conference*, Baltimore, MD, June 30–July 2 2010, pp. 516–523.

[8]Yildiz, Y., Kolmanovsky, I. V., and Acosta, D., "A control allocation system for automatic detection and compensation of phase shift due to actuator rate limiting," *2011 American Control Conference*, San Francisco, CA, June 29–July 1 2011, pp. 444–449.

[9]Yildiz, Y. and Kolmanovsky, I. V., "Stability Properties and Cross-Coupling Performance of the Control Allocation Scheme CAPIO," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 4, 2011, pp. 1190–1196.

[10]Yildiz, Y. and Kolmanovsky, "Implementation of CAPIO for Composite Adaptive Control of Cross-Coupled Unstable Aircraft," *AIAA Infotech*, St. Louis, Missouri, March 29–31 2011, pp. 1–8.

[11]Härkegård, O. and Glad, T., "Resolving actuator redundance–optimal control vs. control allocation," *Automatica*, Vol. 41, No. 1, 2005, pp. 137–144.

[12]Bodson, M., "Evaluation of Optimization Methods for Contrl Allocation," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, August 6–9 2001, pp. 1–15.

[13]Prieto, F. J., *Sequential quadratic programming algorithms for optimization*, Ph.D. thesis, Report SOL 89-7, Department of Operations Research, Stanford University, Stanford, CA, 1989.

[14]Gill, P. E., Murray, W., and Saunders, M. A., "User's guide for QPOPT 1.0: a Fortran package for quadratic programming," Report SOL 95-4, Department of Operations Research, Stanford University, Stanford, CA, 1995.

[15]Gill, P. E., Murray, W., and Saunders, M. A., "User's Guide for SQOPT 5.3: a Fortran Package for Large-Scale Linear and Quadratic Programming," Numerical Analysis Report 97-4, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.

[16]Gill, P. and Wong, E., "Methods for Convex and General Quadratic Programming," Tech. rep., UCSD Department of Mathematics, La Jolla, CA, September 2010, Technical Report NA-10-01.

[17]Moré, J. J. and Toraldo, G., "Algorithms for bound constrained quadratic programming problems," *Numer. Math.*, Vol. 55, 1989, pp. 377–400.

[18]Moré, J. J. and Toraldo, G., "On the solution of large quadratic programming problems with bound constraints," *SIAM J. Optim.*, Vol. 1, No. 1, 1991, pp. 93–113.

[19]Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C., "A limited memory algorithm for bound constrained optimization," *SIAM J. Sci. Comput.*, Vol. 16, 1995, pp. 1190–1208.

[20]Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., "Algorithm 778: L-BFGS-B—Fortran subroutines for large-scale bound constrained optimization," *ACM Trans. Math. Software*, Vol. 23, 1997, pp. 550–560.

[21]Shweyk, K. M., Hyde, D. C., and Levengood, K., "Validation of the flight control system of a conceptual powered-lift, speed-agile, transport aircraft," *AIAA Atmospheric Flight Mechanics Conference*, No. AIAA 2012-4579, Minneapolis, Minnesota, August 13–16 2012, pp. 1–11.

[22]Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, London and New York, 1981, ISBN 0-12-283952-8.

[23]Fletcher, R., *Practical Methods of Optimization*, Volume 2: Constrained Optimization, John Wiley and Sons, Chichester and New York, 1981.

[24]Moré, J. J. and Thuente, D. J., "Line search algorithms with guaranteed sufficient decrease," *ACM Trans. Math. Software*, Vol. 20, 1994, pp. 286–307.

[25]Wolfe, P., "Convergence conditions for ascent methods," *SIAM Review*, Vol. 11, 1968, pp. 226–235.

[26]Powell, M. J. D., "Updating conjugate directions by the BFGS formula," *Math. Prog.*, Vol. 38, 1987, pp. 693–726.

[27]Siegel, D., "Modifying the BFGS update by a new column scaling technique," *Mathematical Programming*, Vol. 66, 1994, pp. 45–78, Ser. A.

[28]Craun, R. W., Acosta, D. M., Beard, S. D., Hardy, G. H., Leonard, M. W., Weinstein, M., and Yildiz, Y., "Motion-Based Piloted Simulation Evaluation of a Control Allocation Technique to Recover from Piloted Induced Oscillations," *AIAA Guidance, Navigation, and Control Conference*, Boston, Massachusetts, August 19–22 2013, p. TBD.