(3)

Sponsored by—
American Institute of Aeronautics and Astronautics (AIAA)
With Participation of—
American Astronautical Society (AAS)

AIAA Paper No. 74-824

SHUTTLE ASCENT TRAJECTORY OPTIMIZATION WITH FUNCTION SPACE QUASI-NEWTON TECHNIQUES

by
E. R. EDGE and W. F. POWERS
The University of Michigan
Ann Arbor, Michigan

9 SEP 1974
MCDONNELL DOUGLAS
RESEARCH & ENGINEERING LIBRARY
ST. LOUIS

AIAA PAPER 74-824

AIAA Mechanics and Control of Flight Conference

ANAHEIM, CALIFORNIA / AUGUST 5-9, 1974

M74-14797

First publication rights reserved by American Institute of Aeronautics and Astronautics. 1290 Avenue of the Americas, New York, N. Y. 10019. Abstracts may be published without permission if credit is given to author and to AIAA. (Price: AIAA Member $1.50. Nonmember $2.00).

Note: This paper available at AIAA New York office for six months; thereafter, photoprint copies are available at photocopy prices from AIAA Library, 750 3rd Avenue, New York, New York 10017

(12)

SHUTTLE ASCENT TRAJECTORY OPTIMIZATION WITH FUNCTION SPACE QUASI-NEWTON TECHNIQUES *

E. R. Edge and W.F. Powers
The University of Michigan
Ann Arbor, Michigan

Abstract

A Space Shuttle ascent trajectory optimization problem from lift-off to orbital insertion is solved with a function space version of a Quasi-Newton parameter optimization method developed by Broyden. The problem includes five parameter and one bounded function controls, two state variable constraints, and four terminal conditions. The bounded controls are treated directly while the remaining constraints are adjoined to the performance index (maximum payload) with penalty functions. The problem is formulated as a four-phase variational problem (liftoff, pitch-over, gravity-turn, linear tangent steering) and the appropriate gradients are developed by first variation theory. A projection operator is introduced to aid in the interpretation of the algorithm with mixed parameter and function controls. Also, by proper partitioning of the computation sequence and storage, storage problems associated with this algorithm are virtually eliminated. The algorithm is applied to the pressure-fed series-burn booster/040C orbiter vehicle and typical simulations are presented. In addition to a discussion of convergence characteristics, the effects of a man-in-the-loop in the optimization process (with a time-shared computer graphics terminal) are presented.

1. INTRODUCTION

A number of function space versions of successful parameter optimization methods have been proposed for optimal control problems in the past few years, especially the function space Davidon method.[1-8] However, in Refs. 1-8, only simple optimal control problems were solved, and even though the convergence properties were good, the storage problems associated with the methods suggested that they might not be applicable to larger-scale problems. Thus, one of the major goals of this research was to apply one of the methods, the function space Broyden method[7], to a nontrivial aerospace trajectory problem requiring considerable storage and numerical integration. As will be discussed later, by proper partitioning of the computation sequence and storage, the drawback of the algorithms due to storage problems is virtually eliminated.

The algorithm is applied to the payload maximization problem for the pressure-fed series-burn shuttle booster/040C orbiter vehicle. Although this is not the current shuttle design, this model was chosen for two reasons: (1) NASA-JSC had considerable data and simulation results for this vehicle when the study was initiated, and (2) it was reported that a singular thrusting arc might exist in the boost stage of the optimal trajectory[9]. Since the function space quasi-Newton algorithms were successful on a number of other problems with bounded controls and singular subarcs[8], the possibility of a singular subarc served as an additional test for the algorithms.

In Section 2, the vehicle, mission constraints, and performance index are defined. In Section 3, the function space Broyden algorithm and associated theory will be presented, while Section 4 presents the important computer implementation aspects of the algorithm. Numerical results are presented in Section 5 and conclusions in Section 6. It should be noted at the outset that this paper is mainly concerned with the study and improvement of the function space quasi-Newton methods, and not with a comparison to other algorithms.

2. VEHICLE AND MISSION

The vehicle and mission considered are taken from Reference 10. The goal is to determine the control history for the pressure-fed series burn shuttle booster/040C orbiter launched from KSC which will yield maximum payload deliverable to a 50 x 100 nm orbit inclined 28.5 degrees. The vehicle is constrained to a non-lifting trajectory with a maximum dynamic pressure of 650 psf and a maximum acceleration of 3.0g's. The trajectory is determined by two controls, the mass flow rate $\dot{m}$, which implies the thrust magnitude, and a thrust angle.

The overall trajectory is subdivided into four "phases" Each of the phases is characterized by the way in which the thrust angle is determined and by the coordinate system in which the equations of motion are being integrated (see Figure 1).

*This work was supported by the National Aeronautics and Space Administration Johnson Space Center under Contract NAS 9-12872 and the National Science Foundation under Grant GK-30115.

FIRST STAGE			SECOND STAGE
COORDINATE SYSTEM SPHERICAL ROTATING			POLAR
PHASE 1	PHASE 2	PHASE 3	PHASE 4
VERTICAL RISE	PITCH OVER	GRAVITY TURN	LINEAR TANGENT

Figure 1. Definition of trajectory phases.

The equations of motion for the first stage are integrated in a spherical coordinate system which rotates with the earth. This coordinate system was chosen because of the ease of representing initial conditions and aerodynamic forces.

Assuming the first stage engines are perfectly expanded to vacuum pressure, the thrust magnitude is

$$|\bar{T}_1| = I_{sp_1} |\dot{m}_1| - P_{atm} A_{exit} \qquad (2.1)$$

The first stage burn is divided into three phases. They are,

i) Phase 1 - vertical rise for ten seconds, $\bar{T} \,||\, \bar{r}$.

ii) Phase 2 - pitch over at a constant rate from vertical and at a constant azimuth angle ψ for 10 seconds(see Figure 2).

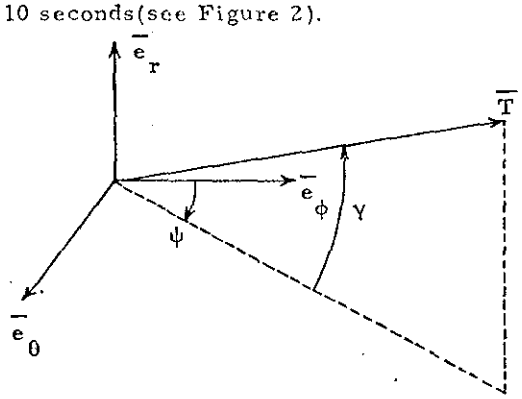

Figure 2. Thrust angles for first stage.

The plane defined by $\bar{e}_\theta$ and $\bar{e}_\phi$ is the local horizontal. The unit vector $\bar{e}_\phi$ points in the easterly direction for $\theta \neq 0$ or π. The vehicle pitches over and at the same time the plane of the orbit is determined by thrusting at a constant azimuth angle ψ. The initial thrust is in the vertical direction, i.e., $\gamma = \frac{\pi}{2}$. The vehicle pitches over with $\dot{\gamma}$ = constant, thus,

$$\gamma = \frac{\pi}{2} - \dot{\gamma}\,(t-10), \quad t \in [10, 20] \qquad (2.2)$$

It is noted that ψ will not correspond to the final inclination. However, the final inclination will be very strongly influenced by ψ and, in fact, ψ will be the primary control which affects the final orbital inclination.

iii) Phase 3 - gravity turn, i.e., the thrust is parallel to the velocity ($\bar{T} \,||\, \bar{V}$). This phase terminates when all fuel is exhausted in the first stage.

Aerodynamic drag is approximately 2% of the total force acting on the vehicle after staging and drops off rapidly thereafter. Thus, aerodynamic forces are neglected during second stage burn. Assuming no out of plane thrust during second stage this allows the equations of motion to be integrated in a polar coordinate system. The change of coordinate systems results in a new set of state variables and a set of transformation equations relating the state after staging to the state before staging. By integrating the equations in a polar coordinate system, the number of state variables is reduced from six to four, and the terminal boundary conditions and adjoint equations are simplified.

The total second stage burn is:

iv) Phase 4 - during second stage burn the thrust is orientated according to the linear tangent steering law, i.e.,

$$\tan\gamma = a\,t + b \;(a, b \text{ constants}) \qquad (2.3)$$

where γ is the angle between the thrust vector and the local horizontal. The second stage engines are perfectly expanded to vacuum pressure, thus,

$$|\bar{T}_2| = I_{sp_2} |\dot{m}_2| \qquad (2.4)$$

This phase terminates when all fuel is exhausted in the second stage.

3. THE OPTIMIZATION PROBLEM AND ALGORITHM

3.1 The Broyden Algorithm in Dyadic Form

A motivating way of viewing the quasi-Newton methods is as a class of algorithms between the first order[12] and second-order[12-15] optimal control gradient methods. The goal of a quasi-Newton algorithm is to build information about the second-variation operator without computing it explicitly, i.e., based upon gradient information only. The Broyden algorithm will be discussed here, however the Davidon, conjugate gradient, and gradient algorithms are easily incorporated into the same computer program, which is the case of the computer program described in Ref. 16.

Consider the general problem:

$$\text{Minimize: } J(u) = \phi(x_f) + \int_{t_o}^{t_f} L(t, x, u)\, dt \qquad (3.1)$$

$$\text{Subject to: } \dot{x} = f(t, x, u), \quad x(t_o) = x_o \; (x \equiv k\text{-vector}) \qquad (3.2)$$

$$|u^i| \leq K_i \; (i = 1, \ldots, m) \; (u \equiv m\text{-vector})$$

$$t_o, t_f \text{ specified}$$

If terminal conditions are present, they are included in the $\phi(x_f)$ - term by the method of penalty functions. In all of the algorithms, the following equations are required:

$$H = L + \lambda^T f\,(t, x, u) \qquad (3.3)$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x}, \quad \lambda(t_f) = \frac{\partial \phi}{\partial x_f} \qquad (3.4)$$

$$g(u) = \frac{\partial H}{\partial u} \qquad (3.5)$$

The function H is the Hamiltonian and $g(u) = \partial H / \partial u$ is the function space gradient.

Each algorithm requires the specification of an initial control $u_o(t)$. In addition, the Broyden and Davidon algorithms require the specification of a positive-definite, self-adjoint linear operator, $\hat{H}_o$, the simplest choice being the identity operator. On each iterate a new control is generated by the update formula,

$$u_{i+1} = u_i + \alpha_i d_i \qquad (3.6)$$

where

$$d_i = \text{search direction} = -\hat{H}_i g_i \qquad (3.7)$$

and α_i = scalar parameter defined by a one-dimensional search technique which minimizes J with respect to α.

The $\hat{H}_i$ operator is updated by

$$\hat{H}_{i+1} = \hat{H}_i + \left[1 + \frac{\langle y_i, \hat{H}_i y_i \rangle}{\langle s_i, y_i \rangle}\right] \frac{s_i \rangle\langle s_i}{\langle s_i, y_i \rangle} - \frac{s_i \rangle\langle \hat{H}_i y_i}{\langle s_i, y_i \rangle} - \frac{\hat{H}_i y_i \rangle\langle s_i}{\langle s_i, y_i \rangle} \qquad (3.8)$$

where

$$s_i = u_{i+1} - u_i \qquad (3.9)$$

$$y_i = g(u_{i+1}) - g(u_i) \qquad (3.10)$$

$$\langle u, v \rangle = \int_{t_o}^{t_f} u^T v \, dt \qquad (3.11)$$

and $u \rangle\langle v$ is an integral kernel dyadic operator such that,

$$(u \rangle\langle v)\, w = \int_{t_o}^{t_f} u(t)\, v(s)^T w(s)\, ds$$
$$= u(t) \int_{t_o}^{t_f} v(s)^T w(s)\, ds \qquad (3.12)$$

The primary difficulty in implementing the quasi-Newton type algorithms on optimal control problems lies in representing the infinite-dimensional integral kernel $\hat{H}$-operator, a function of two variables. One way to overcome this difficulty is to observe that only $\hat{H}_i g_i$ (not $\hat{H}_i$ itself) is needed to compute d_i. Thus to implement the Broyden algorithm, where g is the gradient of a functional, and u, s, and y are time functions, we proceed as follows:

i) Specify $\hat{H}_o$ (any positive definite self-adjoint operator).

ii) Express $\hat{H}_i$ in Eq. (3.8) as a sum back to $\hat{H}_o$. Operate on the resultant expression for $\hat{H}_i$ with g_i *to obtain the following search direction:*

$$d_i = -\hat{H}_o g_i - \sum_{j=0}^{i-1} \left[\left(1 + \frac{\langle y_j, \hat{H}_j y_j \rangle}{\langle s_j, y_j \rangle}\right) \frac{\langle s_j, g_i \rangle}{\langle s_j, y_j \rangle} s_j - \frac{\langle \hat{H}_j y_j, g_i \rangle}{\langle s_j, y_j \rangle} s_j - \frac{\langle s_j, g_i \rangle}{\langle s_j, y_j \rangle} \hat{H}_j y_j \right] \qquad (3.13)$$

Equation (3.13) requires the computation of inner products of the functions $\hat{H}_i y_i$, s_i, and y_i, and operating with $\hat{H}_o$. The functions $(s_o, \ldots, s_{i-1})$ are available from past iterations. To compute the functions $\hat{H}_i y_i$, we need only replace $-g_i$ by y_i in Eq. (3.13), i.e., $\hat{H}_i$ operating on y_i instead of $-g_i$. Then, for the case i-1:

$$\hat{H}_{i-1} y_{i-1} = \hat{H}_o y_{i-1} + \sum_{j=0}^{i-2} \left[\left(1 + \frac{\langle y_j, \hat{H}_j y_j \rangle}{\langle s_j, y_j \rangle}\right) \frac{\langle s_j, y_{i-1} \rangle}{\langle s_j, y_j \rangle} s_j - \frac{\langle \hat{H}_j y_j, y_{i-1} \rangle}{\langle s_j, y_j \rangle} s_j - \frac{\langle s_j, y_{i-1} \rangle}{\langle s_j, y_j \rangle} \hat{H}_j y_j \right] \qquad (3.14)$$

Thus $\hat{H}_{i-1} y_{i-1}$ can be computed in a way requiring only inner products and operation with $\hat{H}_o = I$, as was the case for $-\hat{H}_i g_i$. Note that 2i + 4 time functions must be stored after the i-iteration in order to compute the i + 1 iterate, i.e.,

$(s_o, \ldots, s_i)$	i + 1 functions
$(\hat{H}_o y_o, \ldots, \hat{H}_{i-1} y_{i-1})$	i functions
g_i, u_{i+1}, y_{i-1}	3 functions

Figure 3 shows the flow of the function space Broyden algorithm on a general iterate.

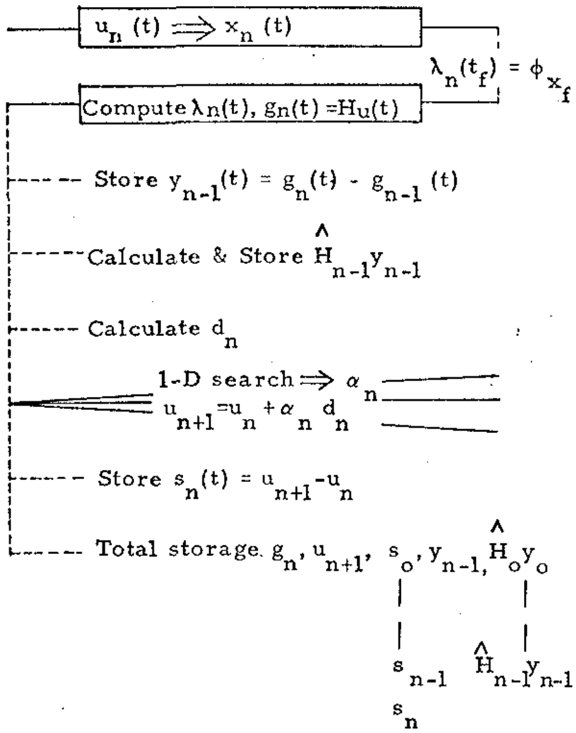

Figure 3. Flow of the Broyden Algorithm

3.2 Function and Parameter Controls

Some optimization problems are most naturally formulated using a combination of function and parameter controls from the product space[17] $L_2^m[t_o, t_f] \times R^n$, and the shuttle ascent optimization is such a problem.

Consider the class of optimal control problems whose control space is

$$\bar{u} \equiv (u_1(t), \ldots, u_m(t);\ c_1, \ldots, c_n) \in L_2^m[t_o, t_f] \times R^n \tag{3.15}$$

where $(u_1(t), \_\_, u_m(t)) \in \quad L_2^m[t_o, t_f]$

$$(c_1, \_\_\_, c_n) \quad \in \quad R^n$$

Upon expansion of the performance index (3.1) about a candidate control and appropriate definitions of adjoint functions (Eq. 3.4), the change in cost is:[12,16]

$$\delta J = \int_{t_o}^{t_f} H_u^T \delta u \, dt + \int_{t_o}^{t_f} H_c^T \delta c \, dt \tag{3.16}$$

Since $c_i \equiv$ constant, then $\delta c \equiv dc$ and Eq. 3.16 becomes

$$\delta J = \int_{t_o}^{t_f} H_u^T \delta u \, dt + dc^T \int_{t_o}^{t_f} H_c \, dt \tag{3.17}$$

The quasi-Newton methods require inner products involving the gradient of the cost with respect to the control. The choice of the inner product must be consistent with existing convergence criteria for the methods.[1] This consistency may be obtained with the projection operator approach which follows.

Note that finite elements in R^n defined on $[t_o, t_f]$ are also elements of $L_2^n[t_o, t_f]$. Thus the control (3.15) may be treated as an element of $L_2^p[t_o, t_f]$ $(p = m + n)$ with a special structure. Then, the admissible control space is a subspace S of $L_2^p[t_o, t_f]$,

$$S = \{\bar{u} \mid u_i(i = 1, \ldots, m) \in L_2^m[t_o, t_f]\ ;\ u_i(i = m+1, \ldots, p) \text{ finite constant time functions.}\} \tag{3.18}$$

In optimal control the linear-quadratic problem (LQP) plays a role similar to the unconstrained quadratic function minimization problem in parameter optimization with respect to the development of properties for quasi-Newton algorithms. The following convergence theorem applies to the LQP, where g is the gradient of the performance index with respect to the control.[1]

<u>Property</u>: Let M be a linear subspace of a Hilbert space D. Let $P: D \to M$ be an operator which is linear, self-adjoint, and idempotent (projection operator). If $\hat{H}_o$ of the quasi-Newton algorithms is chosen to be P and $\bar{u}_o \in M$, then $\bar{u}_i \in M$ for all i, and

$$\lim_{k \to \infty} \| \hat{H}_o g_k \|^2 = 0, \tag{3.19}$$

that is, the projection of the gradient onto M tends to zero (the condition for convergence).

A projection operator $P : L_2^p[t_o, t_f] \to S$ which allows for a consistent method for handling combinations of function and constant type controls is given in the following property, which is proved in Appendix B.

<u>Property</u>: Let $\bar{A} = \begin{bmatrix} A_f(t) \\ \hline A_c(t) \end{bmatrix}$ where $\bar{A} \in L_2^p[t_o, t_f]$

and $A_f(t) \in L_2^m[t_o, t_f]$, $A_c(t) \in L_2^{p-m}$ $[t_o, t_f]$. Define $P: L_2^m[t_o, t_f] \rightarrow S$ by

$$P\bar{A} = P\begin{bmatrix} A_f(t) \\ \hline A_c(t) \end{bmatrix} = \begin{bmatrix} A_f(t) \\ \hline \frac{1}{\Delta T}\int_{t_o}^{t_f} A_c(t)\,dt \end{bmatrix}$$

$$\Delta T = t_f - t_o. \tag{3.20}$$

Then, P is a projection operator.

The above property implies how the first variation (3.17) should be utilized in the quasi-Newton algorithms. First, considering $(u_1(t), \ldots, u_m(t), c_1, \ldots, c_n)$ as an element of $L_2^{m+n}[t_o, t_f]$, the gradient is

$$\tilde{g} = [H_u : H_c] \tag{3.21}$$

and an admissible choice for $\hat{H}_o$, say $\tilde{H}_o$, is the projection operator (3.20), which implies that the initial search direction is

$$\tilde{H}_o\,\tilde{g}_o = \left[H_u^{(o)} : \frac{1}{t_f - t_o}\int_{t_o}^{t_f} H_c^{(o)} dt\right] \tag{3.22}$$

However, note that this is equivalent to assuming

$$g = \left[H_u : \frac{1}{t_f - t_o}\int_{t_o}^{t_f} H_c\,dt\right] \tag{3.23}$$

with $(u_1(t), \ldots, u_m(t), c_1, \ldots, c_n) \in L_2^m[t_o, t_f] \times R^n$, and $\hat{H}_o = I$ since

$$\hat{H}_o g_o = I g_o = \left[H_u^{(o)} : \frac{1}{t_f - t_o}\int_{t_o}^{t_f} H_c^{(o)} dt\right] = \tilde{H}_o \tilde{g}_o \tag{3.24}$$

Furthermore, the choice of definition for the gradient (3.23) has the same convergence properties as the choice (3.21) since

$$\|\tilde{H}_o \tilde{g}_k\| \to 0$$

implies, with $\hat{H}_o = I$,

$$\|\hat{H}_o g_k\| = \|g_k\| = \|\tilde{H}_o \tilde{g}_k\| \to 0. \tag{3.25}$$

For convenience, Eq. (3.23) will be utilized as the gradient expression.

3.3 The First Variation

For convenience let u denote the total control vector $(C_1, \ldots, C_5, |\dot{m}|)$. The equations of motion may be symbolized by

$$\begin{aligned} \dot{x} &= f(t, x, u) & t &\in [0, t_s) \\ \dot{\tilde{x}} &= \tilde{f}(t, \tilde{x}, u) & t &\in [t_s, t_f] \end{aligned} \tag{3.26}$$

where t_s and t_f are the staging and final times, respectively. At t_s the states are related by the transformation equation,

$$\tilde{x}(t_s^+) = g(x(t_s^-)) \tag{3.27}$$

The terminal boundary conditions are handled by the method of quadratic penalty functions, and the state variable inequality constraints are handled by integral quadratic penalty functions. The performance index is,

$$\begin{aligned} J(u) = {} & -m_o + P_1(\tilde{x}_1(t_f) - \tilde{x}_{1f})^2 + P_2(\tilde{x}_2(t_f) - \tilde{x}_{2f})^2 + P_3(\tilde{x}_3(t_f) - \tilde{x}_{3f})^2 \\ & + P_4 \int_{to}^{t_s^-} (q-650)^2\, U(q-650)\,dt \\ & + P_5 \int_{to}^{t_s^-} (acc-3.00)^2\, U(acc-3.00)\,dt \\ & + P_6 \int_{t_s^+}^{t_f} (acc-3.00)^2\, U(acc-3.00)\,dt \\ & + P_7 (\cos\Phi(t_s) - \cos\Phi_f)^2. \end{aligned} \tag{3.28}$$

Then, the following multistage optimal control problem is defined:

$$\text{Min. } J(u) = \phi(x_o, x_s, \tilde{x}_f) + \int_0^{10-} L_1(t, x, u)\,dt + \int_{10+}^{20-} L_2(t, x, u)\,dt + \int_{20+}^{t_s^-} L_3(t, x, u)\,dt + \int_{t_s^+}^{t_f} L_4(t, \tilde{x}, u)\,dt \tag{3.29}$$

$$\text{Subject to: } \dot{x} = f(t, x, u) \quad (t_o \le t < t_s)$$

$$\text{and} \quad \dot{\tilde{x}} = \tilde{f}(t, \tilde{x}, u) \quad (t_s < t \le t_f) \tag{3.30}$$

With

$$\begin{aligned} H &= L + \lambda^T f & &\text{on } [t_o, t_s) \\ \tilde{H} &= \tilde{L} + \tilde{\lambda}^T \tilde{f} & &\text{on } [t_s, t_f], \end{aligned} \tag{3.31}$$

the adjoint equations and associated boundary conditions are (see Ref. 12 or 16):

$$\begin{aligned} \dot{\lambda} &= -\frac{\partial H}{\partial x} & &\text{on } [t_o, 10), (10, 20), (20, t_s) \\ \dot{\tilde{\lambda}} &= -\frac{\partial \tilde{H}}{\partial \tilde{x}} & &\text{on } [t_s, t_f] \\ \lambda_i(t_f) &= \phi_{x_{if}} & & i = 1, 2, 3 \\ \tilde{H}(t_f) &= 0 \implies \text{equation for } \tilde{\lambda}_4(t_f) \\ \lambda(t_s^-) &= \frac{\partial g}{\partial x}\Big|_{t_s}^T \tilde{\lambda}(t_s^+) + \phi_{x_s} \\ H(t_s^-) &= \tilde{H}(t_s^+) \implies \lambda_6(t_s^-). \end{aligned} \tag{3.32}$$

TABLE 1. MISSION TIMETABLE

$t_o = 0$ sec	10 sec	20 sec	t_s^- free	t_s^+ free	t_f free
x_1-x_5 fixed	free	free	x_1-x_5 free	$\tilde{x}(t_s^+) = g(x(t_s^-))$	$\tilde{x}_1$-$\tilde{x}_3$ free
x_6 free			$\psi_s(x_6) = 0$	↑	$\psi_f(\tilde{x}_4) = 0$
			↑ Mass of Fuel 1st stage depleted defines t_s	Transformation Equations	↑ Mass of fuel 2nd stage depleted defines t_f

Then, the change in cost due to δu and dm_o is:

$$\delta J = [\phi_{x_{60}} + \lambda_6(t_o) - \lambda_6(t_s^-) + \hat{\lambda}_4(t_s^+) - \tilde{\lambda}_4(t_f)]\, dm_o$$
$$+ \int_0^{10-} H_u^T \delta u\, dt + \int_{+10}^{20-} H_u^T \delta u\, dt + \int_{20+}^{t_s^-} H_u \delta u\, dt$$
$$+ \int_{t_s^+}^{t_f} \tilde{H}_u^T \delta u\, dt \qquad (3.33)$$

The particular choices for $dm_o \equiv m_o^{(n+1)} - m_o^{(n)}$, $\delta u(t) \equiv u^{(n+1)} - u^{(n)}(t)$, for the $n+1$ iterate, are governed by the choice of algorithm.

We now wish to interpret Eq. (3.33) for use in the Broyden method. Noting the form of Eq. (3.17), we rewrite the first term of Eq. (3.33) as

$$dm_o \int_{t_o}^{t_f} A/(t_f - t_o)\, dt, \qquad (3.34)$$

where A is the coefficient of dm_o in Eq. (3.33). Then, the gradient corresponding to Eq. (3.23) is:

$$g = \left[\frac{1}{t_f - t_o} \int_{t_o}^{t_f} A/(t_f - t_o)\, dt \,\middle|\, \frac{1}{t_f - t_o} \int_{t_o}^{t_f} H_C\, dt \,\middle|\, H_{u_6}\right] \qquad (3.35)$$

4. COMPUTER IMPLEMENTATION

4.1 Computer Graphics Aspects

Figure 4 is the flow diagram of the shuttle ascent trajectory optimization program. The main iteration loop consists of the forward integration, backward integration, calculation of search direction, 1-D search, and convergence check. These operations are repeated for a given set of penalty coefficients until **an** "acceptable" degree of convergence is obtained. At this point the human operator interrupts the executing program.

Because the terminal boundary conditions and state variable inequality constraints are

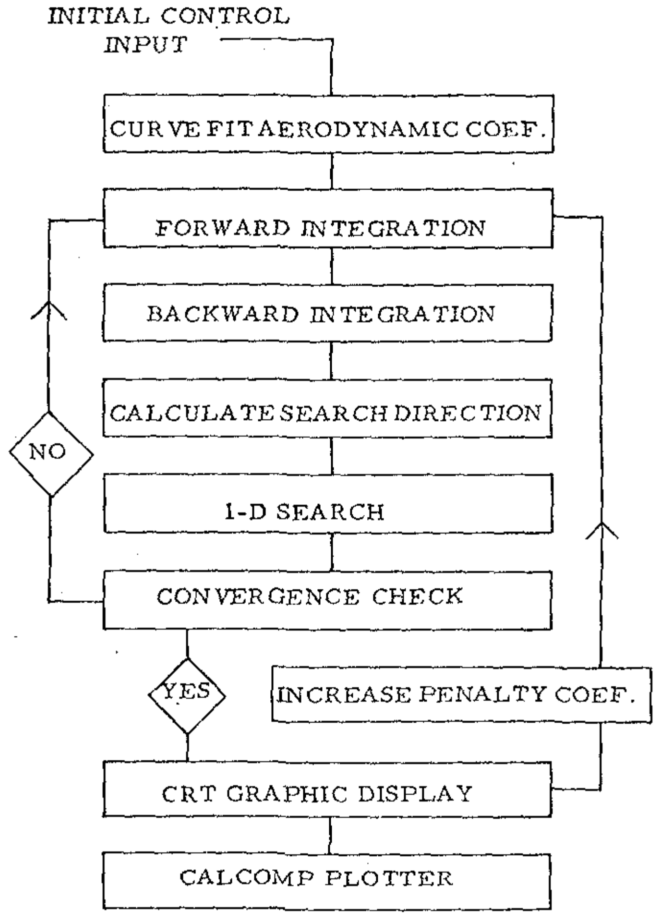

Figure 4. Flow of Computer Program

handled by penalty methods it has been found that a considerable savings in computer time can be achieved by real time human interaction with the executing program. Recall that P_i (i=1, 2, 3, 7) are the penalty coefficients associated with the terminal boundary conditions and P_i (i=4, 5, 6) are the penalty coefficients associated with the state variable inequality constraints. For a given set of penalty coefficients a particular unconstrained optimization problem is defined. The solution to the original constrained optimization problem is approximated by a sequence of solutions to the unconstrained problem generated by letting P_i (i=1, . . . , 7) $\rightarrow \infty$. As P_i (i=1, 2, 3, 7) are increased the solutions generated will more closely satisfy the requirements of a 50 x 100 nm orbit inclined

28.5° to the equator entered at perigee. Likewise as P_i (i=4, 5, 6) are increased the state variable inequality constraints on dynamic pressure and acceleration are more strictly enforced. The ultimate goal is to find the control history which yields the maximum liftoff weight and satisfies all seven of the constraints. As expected, in practice as one penalty coefficient is increased the error associated with it will decrease while the errors associated with the other coefficients will increase. Thus by improving the trajectory in one respect it is possible to lose something somewhere else. Sensitivity to changes in the different penalty coefficients also varies. As the penalty coefficients become larger the overall problem will become increasingly sensitive to changes in the control and numerical instability will eventually result. The way in which the penalty coefficients are increased will strongly influence the overall convergence rate of the algorithms. The main drawback to the method of penalty functions is that the penalty coefficients must be increased in a problem dependent way. Even for simple example problems which require little computer time for a trajectory integration and which have only one or two penalty coefficients, the choice of these coefficients and the way in which they are increased is critical for rapid convergence. Because of the complexity and relatively long computer time required for a trajectory integration of the shuttle ascent optimization problem, it is desirable to have rapid feedback of the progress of the algorithm.

By using time sharing computers and CRT display terminals the problem of choosing penalty coefficient values can be very efficiently solved by human operator interaction with the executing program. At the end of a specified number of iterations, execution is terminated and control transfered to a CRT display terminal. Because of time sharing this interruption of the executing program is very inexpensive. At the request of the human operator important information is then graphically displayed on the CRT. The information is evaluated and a decision on changes of the penalty coefficients is reached. This information is communicated to the computer and execution proceeds. By placing a human operator in the program iteration cycle convergence times are reduced, the computer is used more efficiently, and the operator quickly builds an intuitive feel for the physical problem being solved.

For the shuttle ascent optimization problem it is helpful to graphically display dynamic pressure, acceleration, and $\dot{m}$ as functions of time along with terminal miss values. The best convergence rate was achieved by first increasing P_i(i=1, 2, 3, 7) yielding a trajectory which comes "close" to the desired terminal boundary conditions. Then P_i (i=4, 5, 6) are increased to enforce the state variable inequality constraints while simultaneously increasing P_i(i=1, 2, 3, 7) so that all intermediate trajectories remain "close" to the terminal boundary conditions.

The ability to interact with the executing program can be useful in other ways, e.g., the interrelationship of adjoint, state variable, search direction, and gradient time histories can be conveniently analyzed using the CRT display. In conclusion, the ability to communicate with the executing program is a valuable tool for solving large-scale optimization problems.

4.2 Storage Problems With Quasi-Newton Algorithms

It was shown in Section 3 that 2 i + 4 time functions must be stored after the i^{th} iterate in order to compute the i+1 search direction. Each of these functions is stored as a vector of numbers which correspond to the function values at N equally spaced points on $[t_o, t_f]$. Thus $(2i + 4) \times N$ floating point numbers must be stored after the i^{th} iterate. The computation per iterate also increases because of the increased number of inner product evaluations. Thus, in the past, it has been a practical necessity to restart the algorithms to a pure gradient step every q^{th} iterate. It has been found[5] that $3 < q < 8$ appears to be a good choice. The value of N must be large enough so that a "good" representation of the functions is obtained. For the shuttle optimization problem the time interval is approximately 500 seconds and N was chosen to be 500. Thus storage must be allocated for $(2q + 4)\ N = (2 \times 8 + 4)\ 500 = 10{,}000$ double precision floating point numbers. Additional storage must be allocated for other variables used in the program and for the object program.

During the initial testing of the program on the University of Michigan IBM 360/67 virtual memory computer all storage was done in fast memory. However, core storage was exceeded on the initial simulations on the JSC's Univac 1108 computer. To overcome this difficulty the 10, 000 double precision floating point numbers needed for the quasi-Newton algorithms were placed on drum storage. This reduced the amount of core storage required allowing the program to fit on the 1108. Upon running the modified program on the IBM computer a considerable savings was realized in reduced virtual memory charges. It was also found that no significant increase in the amount of CPU time was incurred. There are two reasons for this:

i) A very small percent of CPU time is spent calculating the search direction. Most of the CPU time is spent integrating the equations of motion. (On each iterate a forward integration and a backward integration are required to determine the gradient and a number of cost evaluations also requiring forward integrations are performed by the 1-D search.)

ii) The updating equation for $H_i y_i$ and the equation for d_i are summations which require inner products of the stored functions in the same sequence as they are generated and stored. For example, assume $H_{i-1} y_{i-1}$ and d_i are to be calculated. $H_o y_o$ through $H_{i-2} y_{i-2}$ are stored in a file sequentially, and the read-write pointer is at $H_o y_o$ (the file is rewound after each iteration). The updating equations for $H_{i-1} y_{i-1}$ will read $H_o y_o, H_1 y_1, \ldots, H_{i-2} y_{i-2}$ in order, calculate $H_{i-1} y_{i-1}$, then write $H_{i-1} y_{i-1}$ onto the file and rewind. Concurrently the equation for d_i has been using the Hy functions. The files in which Hy and s are stored need only be rewound once on a given iteration and no forward or back spacing is required. Even if tape were to be used as the storage medium, instead of fast core storage, the increase in computer time would be small. When drum storage is used the increase in computer time is insignificant. Thus there is no need to reset to a gradient step because of limited storage. However, one may still wish to reset because of round-off error buildup.

As mentioned previously the computation time per iterate increases due to the increasing number of inner product evaluations which must be made. However, since the inner product is a quadrature

$$< u, v > = \int_{t_o}^{t_f} u^T v \, dt \qquad (4.1)$$

where u and v are stored pointwise, if it is assumed that the stored functions are linear between storage locations the evaluation of the inner product is easily reduced to a summation. It was found that this method of evaluating inner products is considerably faster then higher order quadrature formulas and that convergence rates of the algorithms do not suffer.

Another observation which saves computer time and effort is the fact that the control $u_6(t) \equiv |\dot{m}(t)|$ is treated as a piecewise linear function of time. This not only allows for an analytical integration for m (t), but also for the determination of t_s and t_f before each integration since t_s and t_f are defined implicitly by fuel depletion. This avoids the problems of checking for fuel exhaustion at each integration point, and of treating t_f as an optimization parameter (which then requires extension or contraction of the control guess if the mass of propellent is not zero at the guessed t_f).

5. NUMERICAL RESULTS

In Table 2, payload, terminal miss values, and penalty weighting coefficients versus iterate are shown. The initial control is C_1 = payload = 90,000 lbm, $C_2 = \dot{\gamma} = .5028\ ^o$/sec., C_3 = a = $-.3390 \times 10^{-3}$, C_4 = b = 0.3664, $C_5 = \psi = -19.012^o$, and $\dot{m}(t)$ = 98%. This control produced a trajectory with the following terminal errors: Δr = -5317 ft., Δu = 357 fps, Δv = 27.3 fps, and $\Delta\Phi = 2.24^o$. The staging time is 118.7 sec and the final time is 503.3 sec. The state variable inequality constraints (SVIC) are violated; Q reached a peak value of 792 psf at 66.8 sec. while the maximum acceleration during first stage was 3.8 g's and during second stage 3.9 g's.

On the first six iterations the penalty values cause the terminal errors and the SVICs to be enforced roughly equally. Evaluation of the trajectory after the sixth iterate indicated some throttling to enforce the acceleration constraints but little throttling in the region of the dynamic pressure constraint (Figure 5). Thus P_4, the dynamic pressure penalty coefficient is increased. The final condition penalty coefficients, P_1, P_2, and P_3, are also increased so that the terminal errors do not become too large. After the tenth iterate the SVICs are approximately enforced; however the terminal errors are too large. Thus P_5 and P_6 are reduced while increasing P_1, P_2, and P_3. The control history after the thirteenth iterate, Figure 5, causes the SVIC and terminal boundary conditions to be enforced. However, the payload is still increasing and Δr = -2 miles. On iterates fourteen through sixteen the SVIC penalty coefficients are again increased, producing a sharper throttle history. Finally, on iterates seventeen through twenty-three the terminal errors are forced to within acceptable tolerances. On the final trajectory, the staging time is 122.02 sec., and the final time is 502.7 sec., and $\Phi(t_f)$ = 28.8^o. Figure 6 shows the final dynamic pressure and acceleration histories.

6. CONCLUSIONS

A space shuttle ascent trajectory optimization problem is solved with the function space Broyden method. The trajectory consists of four distinct phases (lift-off, pitch-over, gravity-turn, linear tangent steering) with one bounded function control (mass-flow rate) and five parameter controls, one of which is bounded. Penalty functions are employed for two state variable inequality constraints (dynamic pressure and axial acceleration) and the orbital insertion terminal boundary conditions.

A major aspect of the study is the application of a function space quasi-Newton method to a realistic aerospace trajectory optimization problem. To overcome the inherent storage problems of such methods, it is shown that the various inner product calculations can be sequenced and stored in such a way that "slow" storage can efficiently handle the task. In addition, a projection operator is developed which allows for a consistent method for treating problems with both function and parameter controls, where the

TABLE 2. NUMERICAL RESULTS												
--------CRT EVALUATION					$P_i = 0.1 \times 10^{E_i}$							
* GRADIENT STEP					r	u	v	Q	ACC_1	ACC_2	Φ	
ITERATE	PAYLOAD	Δr	Δu	Δv	E_1	E_2	E_3	E_4	E_5	E_6	E_7
*1	91,672	-17,740	269	54	13	15	15	2	16	13	15
2	91,680	-22,100	82	104							
3	91,681	-11,080	21	118							
4	91,686	-462	66	95							
5	91,720	26,610	59	83							
6	91,853	4,947	-5.4	61							
*7	102,976	-18,750	-21	-8.3	14	16	18	4	16	13	15
8	102,946	-16,530	-18	-2.8							
9	102,946	-16,530	-18	-2.7							
10	102,946	-15,510	-17	-2.4							
*11	101,209	-10,950	20	19	17	20	20	4	13	10	10
12	101,209	-11,130	12	22							
13	101,312	-11,070	8.2	23							
*14	101,278	-10,300	165	22	15	17	18	4	16	13	15
15	101,127	-6,403	117	16							
16	101,207	-8,665	47	16							
*17	104,200	-35,000	-98	-60	16	20	20	4	13	10	12
18	104,806	-27,000	-42	-26							
19	105,322	-8,761	-22	-13							
20	105,721	-4,760	-19	-12							
*21	106,206	-2,740	-6	-8.2	18	22	21	3	12	10	11
22	106,606	-861	-4.6	-3.2							
23	106,646	-231	- .7	1.3							

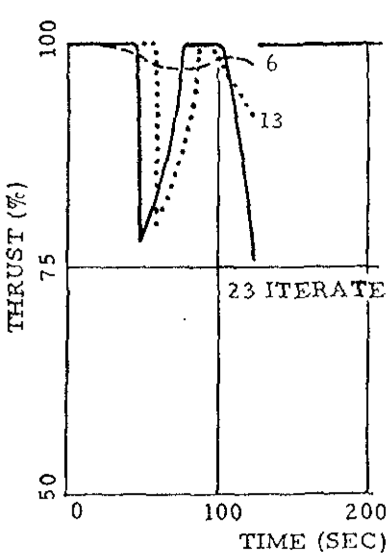

Figure 5. Thrust ($\dot{m}$) History

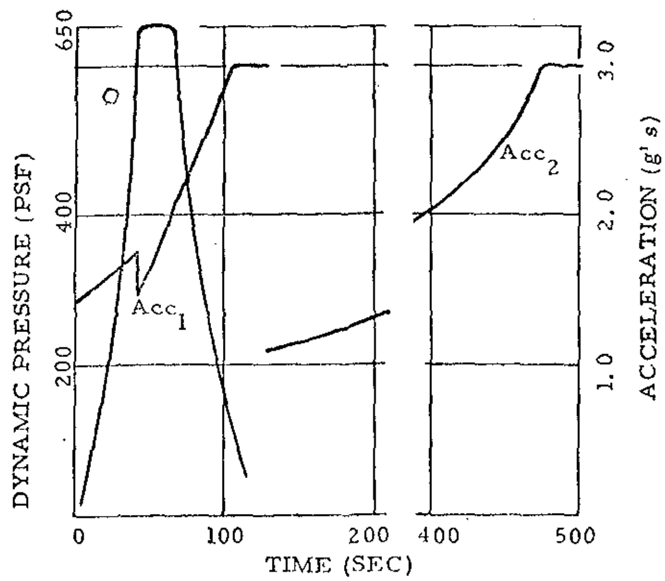

Figure 6. Final Q & Acceleration History

definition of the various mixed inner products is not straight forward. Of course, the same operator is also applicable to other function space quasi-Newton methods (e.g., Davidon, projected gradient).

Because of the relatively large number of penalty coefficients (seven) it was found that a time-shared, interactive graphics capability enhanced considerably the rate of convergence of the problem. The advantages of such a capability are: fewer iterates are "wasted", one learns more about the problem by staying with it on the terminal as opposed to frequent batch-job submissions, and the problem is usually solved much more quickly (e.g., on a time-shared computer, twenty thirty-second runs per hour are feasible whereas one ten minute run in the batch mode usually involves a turn-around time of several hours).

Finally, with respect to the use of the function space quasi-Newton methods, one can see by Figure 3 that the only additional programming (compared to the standard gradient method) involves the inner products in Eqs. 3.13 and 3.14. Whether or not one wishes to do this additional work is, of course, problem dependent (it may be a necessity in problems where the gradient method has convergence problems, e.g., singular problems). However, just as the finite-dimensional space quasi-Newton algorithms have become the major parameter optimization methods in recent years, because of deficiencies in the gradient and Newton methods, a similar situation may occur in optimal control problems with their function-space analogs.

REFERENCES

1. Horwitz, L.B., and Sarachik, P.E., "Davidon's Method in Hilbert Space," SIAM J. Appl. Math., Vol. 16, pp 676-695, 1968.

2. Ladd, H.O., "A Unified Theory for Constrained Minimization on Hilbert Space,"PhD Thesis, Massachusetts Institute of Technoloby, 1968.

3. Lasdon, L.S., "Conjugate Direction Methods for Optimal Control", IEEE Trans. on Auto. Control, Vol. AC-15, pp 267-268, 1970.

4. Tripathi, S.S., and Narendra, K.S., "Optimization Using Conjugate Gradient Methods," IEEE Trans. on Auto Control, Vol. AC-15, pp 268-270, 1970.

5. Pierson, B.L., and Rajtora, S.G., "Computational Experience with the Davidon Method Applied to Optimal Control Problems," IEEE Trans. on Sys. Sci. and Cyber., pp 240-242, 1970.

6. Tokumura, H., Adachi, N., and Goto, K., "Davidon's Method for Minimization Problems in Hilbert Space with an Application to Control Problems," SIAM J. on Control, Vol. 8, pp 163-178, 1970.

7. Edge, E.R., and Powers, W.F., "Function Space Quasi-Newton Algorithms for Optimal Control Problems with Bounded Controls and Singular Arcs," AAS/AIAA Astrodynamics Conference, Vail, Colorado, July 1973.

8. Oi, K., Sayama H., and Takamatsu T., "Computational Schemes of the Davidon-Fletcher-Powell Method in Infinite-Dimensional Space," J. of Optimization Theory and Applications", Vol. 12, No. 5, pp 447-458, 1973.

9. Kamm, J.L., "A Series Burn Booster Throttle Strategy to Increase Payload by 14,000 Pounds," TRW Interoffice Correspondence 4.913.7 - 73 - 34, March 13, 1972.

10. Laszlo, W.O., and Sullivan,H.C.,"Optimal Shuttle Ascent Trajectories Using the PEACE Program," NASA-MSC Memorandum FM 94 (72-82), June 16, 1972.

11. Kamm, J.L., and Kim,I.J., "A Fast and Precise 1963 Patrick Atmosphere Analytical Model," TRW-Houston IOC 5521. 5-13, December 2, 1970.

12. Bryson, A.E., Jr., and Ho, Y.C., Applied Optimal Control, Blaisdell, Waltham,Mass., Chap. 7, 1969.

13. Kelley, H.J., Uzzell, B.R., and McKay,S. S., "Rocket Trajectory Optimization by a Second-Order Numerical Technique," AIAA J., Vol. 7, pp 879-884, 1969.

14. Mitter, S.K., "Successive Approximation Methods for the Solution of Optimal Control Problems," Automatica, Vol. 3, pp 135-149, 1966.

15. Jacobson, D.H., and Mayne, D.Q., Differential Dynamic Programming, American Elsevier, New York, 1970.

16. Edge, E.R., Shieh, C.J., Powers, W.F., "Techniques For Shuttle Trajectory Optimization," Final Report, NASA Contract NAS 9-12872, December, 1973.

17. Porter, W.A., Modern Foundations of Systems Engineering, Macmillan, New York, Chap. 3, 1966.

APPENDIX A

Mission Data

In summary the mission constraints and controls are,

i) *Initial conditions*-launch from KSC.

ii) Terminal conditions - 50 x 100 nm orbit inclined 28.5 degrees with insertion at perigee.

iii) Function Type Control - $|\dot{m}|$ mass flow rate, a function of time.

iv) Parameter Type Controls

C_1 - GLOW (Gross Liftoff Weight).

C_2 - $\dot{\gamma}$, pitch-over rate during phase 2.

C_3 - a (linear tangent parameter).

C_4 - b (linear tangent parameter).

C_5 - ψ, out of plane thrust angle during phase 2.

v) State variable inequality constraints - Dynamic Pressure $\leq$ 650 psf. $\text{Acceleration}_{max} \leq$ 3.0 g's

vi) Performance Index - maximize the gross liftoff weight, GLOW.

The mass of the vehicle is broken down into five parts,

i) m_{1f} = fuel first stage = 3.50680×10^6 lbm

ii) m_{1s} = structure first stage = 5.70850×10^5 lbm

iii) m_{2f} = fuel second stage = 1.16415×10^6 lbm

iv) m_{2s} = structure second stage = 2.61300×10^5 lbm

v) m_p = payload = to be maximized

The engines are characterized by,

I_{sp_1} = 270.7 sec. $\qquad A_{exit_1} = 700 \text{ ft}^2$

I_{sp_2} = 456.5 sec.

APPENDIX B

Projection Operator Proof

The operator defined by Eq. (3.20) is a projection operator if it is linear, self-adjoint, and idempotent. These properties will now be proved.

i) *Linearity*:

$$P[\alpha \bar{A} + \beta \bar{B}] = P\begin{bmatrix} \alpha A_f + \beta B_f \\ \hline \alpha A_c + \beta B_c \end{bmatrix}$$

$$= \begin{bmatrix} \alpha A_f + \beta B_f \\ \hline \frac{1}{\Delta T} \int_{t_o}^{t_f} (\alpha A_c + \beta B_c)\, dt \end{bmatrix} = \alpha P\bar{A} + \beta P\bar{B}$$

ii) Self-Adjoint: Define P^* by $\langle \bar{A}, P\bar{B}\rangle = \langle P^*\bar{A}, \bar{B}\rangle$

$$\langle \bar{A}, P\bar{B}\rangle = \int_{t_o}^{t_f} \bar{A}^T P \bar{B}\, dt = \int_{t_o}^{f} [A_f A_c] P \begin{bmatrix} B_f \\ \hline B_c \end{bmatrix} dt$$

$$= \langle P\bar{A}, \bar{B}\rangle \Longrightarrow P^* = P.$$

iii) Idempotent ($P^2 = P$):

$$P[A_f | A_c] = [A_f | \frac{1}{\Delta T} \int_{t_o}^{t_f} A_c\, dt]$$

$$\Longrightarrow P^2 = P$$

LIST OF SYMBOLS

Acc - axial acceleration
A_{exit} - exit area of first stage engines
d_i - search direction
g_i - gradient
H - Hamiltonian
$\hat{H}_i$ - linear operator in quasi-Newton algorithms
I_{sp} - specific impulse
J - performance index
P_{atm} - atmospheric pressure
P_i - seven penalty weighting coefficients
q - dynamic pressure $= \frac{1}{2}\rho V^2$
r - radius
s_i - Δu_i
T - thrust
t_s - staging time
t_f - final time
u - control vector
$U(\cdot) = \begin{cases} 0 & \cdot \leq 0 \\ 1 & \cdot > 0 \end{cases}$
V - velocity
x_1 - first stage radius
x_2 - first stage θ
x_3 - first stage radial velocity
x_4 - first stage tangential velocity
x_5 - first stage normal velocity
x_6 - first stage mass
$\tilde{x}_1$ - second stage radius
$\tilde{x}_2$ - second stage radial velocity
$\tilde{x}_3$ - second stage tangential velocity
$\tilde{x}_4$ - second stage mass
y_i - Δg_i
γ - angle of thrust above local horizontal
θ - spherical coordinate
λ - influence functions
ρ - atmospheric density
ϕ - spherical coordinate
Φ - inclination
ψ - azimuth angle

18 AUG 1974
RESEARCH C...
ST. LOUIS