ABSTRACT

This research develops a general method that combines optimization methods and an interior ballistics model to automate the design process for propellent grains. It is a multi-variable constrained optimization problem. The augmented Lagrange multiplier method is used to control the constrained problem while two zero-order methods (Powell's and Hooke-Jeeves) perform the unconstrained minimization. The interior ballistics model IBRGAC, developed at the Interior Ballistics Laboratory, Aberdeen Proving Grounds, Maryland, is used as the objective cost function. To validate the process a representative 120mm tank gun system is used with four propellent combinations. The examples demonstrate that the scheme works and can be used as an effective design tool.

INTERIOR BALLISTICS OPTIMIZATION

by

JOE ROBERT GONZALEZ JR.

B.S., United States Military Academy, 1978

A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1990

Approved by:

Major Professor

TABLE OF CONTENTS

LIST OF SYMBOLS

Term:	Definition:	Units:
A	Area	m^2
a	Acceleration	m/s^2
b	Covolume	m^3/kg
C	Initial mass of propellent (or igniter)	kg
c_v	Specific heat at constant volume	J/kg-K
c_p	Specific heat at constant pressure	J/kg-K
D	Diameter	m
d	Distance between perforation centers	cm
E	Energy	J
F	Force	N
f	Fraction of work done against bore friction that preheats chamber	none
h	Heat transfer coefficient	$watt/m^2$-K
L	Length of propellent	cm
m	Mass	kg
n	Factor of safety	none
P	Pressure	Pa
p_i	Diameter of inner perforation	cm
p_o	Diameter of outer perforation	cm
$\dot{Q}$	Heat Flow	watts
R	Specific gas constant	kJ/kg-K
$\bar{R}$	Universal Gas Constant	kJ/Kmol-K
r	Linear burning rate	m/s
S	Surface area of partially burned propellent grain	m^2
T	Temperature	K
T_o	Adiabatic flame temperature	K
t	Time	s
V	Volume	m^3
V	Molar volume	m^3/kmol
v	Velocity	m/s
w_i	Distance (web) between inner and outer perforations	cm
w	Distance (web) between outer perforations	cm
w_o	Distance (web) between outer perforations and outer propellent diameter	cm
$\bar{X}$	Design vector (components are numerically subscripted)	none
x	Projectile travel	m
x	Gas distance from breech	m
y	Position of projectile base	m
z	Fraction of mass burned	none
α	Burning rate exponent	none
β	Burning rate coefficient	m/s-Pa
γ	Ratio of specific heats	none

λ	Lagrange multiplier (Except in Chapter IV where it is the Nordheim friction factor.)	none
ρ	density	kg/m^3
σ_{yp}	Yield point strength	MPa

Subscripts:

b	base of projectile
br	breech
c	chamber
g	air pressure in front of projectile
I	igniter propellent
o	original value
p	projectile
rp	recoiling parts of gun
r	bore resistance
t	total of igniter and propellent
w	chamber wall property

CHAPTER I

INTRODUCTION

The advancing capabilities of digital computers has allowed numerical optimization to develop into an effective and useful analysis and design tool for the engineer. It is applied successfully to many design problems in various disciplines. Optimization schemes allow the engineer to evaluate a large number of design alternatives in a systematic and efficient manner to find the best design. The approach is used to improve performance and/or decrease cost while meeting the constraints appropriate to the problem.

Interior ballistics is the applied physics required to impart motion to a projectile inside a gun tube. Despite its long history and wide application, active research continues with efforts to build more realistic models of the complicated chemical, thermodynamic, and dynamic processes involved. The classic interior ballistics problem is: given the characteristics of the gun, charge, and projectile determine the muzzle velocity of the projectile and the peak pressure in the gun. There is a large knowledge base of both theoretically sound and experimentally proven concepts that make the solution of the interior ballistics problem possible.

The burning of chemical compounds, called propellents, is an important part of interior ballistics. The combustion processes depend on many factors including the propellent material and its geometry, the rate of burn, propellent packing and packaging, and environmental factors. The resulting gases produce the pressure field that imparts acceleration to the projectile. The design of the propellent grain shape to achieve the desired pressure-time history, given the constraints imposed by the gun system and projectile, is part of the design effort. The purpose is usually to maximize the projectile muzzle velocity.

Presently there are a number of computer based interior ballistic models with a wide range of capabilities. These models allow the interior ballistician to predict the performance of a particular gun, charge, and projectile combination.

Interior ballistics depends on parameters and variables so numerous that a complete initial investigation of them is not practical. To demonstrate that optimization can be successfully applied, a specific gun system is chosen as an example. The example used is the optimum design of the propellent grain geometry for kinetic energy projectiles that are fired from 120mm tank cannons. The propellent grain is considered improved if there is a net increase in muzzle velocity without violating gun constraints. This is done by application of numerical optimization in conjunction with the interior ballistic model.

The approach taken in this thesis is to use an interior ballistic model, and combine it with an efficient and easy to use optimization method that searches the design space for the maximum muzzle velocity without violating the constraints of the problem.

The optimization method used is a sequential unconstrained minimization technique (SUMT) called the augmented Lagrange multiplier method (ALM). In the version of the ALM used in this thesis the unconstrained minimization is done by Powell's method or the Hooke-Jeeves method. The interior ballistics model used is IBRGAC, developed at the Ballistic Research Laboratory, Aberdeen Proving Grounds, Maryland.

A background section that provides the information necessary for understanding the problem is provided in Chapter 2. This covers optimization, interior ballistics, and gun nomenclature. Chapter 3 details the optimization process and describes the algorithms and computer code used. Chapter 4 outlines the laws, theories, assumptions, and equations used to solve the interior ballistic problem. At the end of the chapter is a description of the interior ballistics code used. Chapter 5 contains the example problems, results, and an analysis of the process. Chapter 6 discusses the conclusions and recommendations for further research.

The optimization scheme is not limited to the particular example considered but has general applicability in interior ballistics, as well as ballistics in general. Although the scope of the problem in this thesis is restricted, the method is not.

CHAPTER II

BACKGROUND

This chapter provides the background to understand the contextual scope and contribution of the thesis. It is comprised of three parts. Part one covers optimization and engineering design. Part two covers ballistics and discusses currently available interior ballistics models. Part three gives a brief background of how gun systems function along with terminology.

1. Optimization and Design.

Optimization is part of human nature. There is no endeavor that man has attempted that he has not tried to improve. Mathematically the problem of finding the extrema of functions, by hand calculation, has a long history (4). The development of the digital computer provided the impetus for the full scale development of numerical optimization. Since Davidon introduced variable-metric methods in 1959 (8) there has been an explosion in the development of optimization schemes, resulting in dozens of reliable, efficient algorithms.

In engineering design, the goal is to produce the "best" design for the desired system or component. The purpose of numerical optimization is to provide a tool to aid the engineer in this task. Engineering problems are normally not confined to one design variable nor is the design space infinite. This results in multi-variable design problems with constraints. The general form for a nonlinear constrained optimization problem can be stated as (17)

Minimize: $F(\overline{X})$..............objective function,

Subject to:

$g_j(\overline{X}) \leq 0$ $\quad$ $j=1,n$..........inequality constraints

$h_k(\overline{X}) = 0$ $\quad$ $k=1,l$..........equality constraints

$x_i^{(lower)} \leq x_i \leq x_i^{(upper)}$ $\quad$ side constraints

$i=1,n,$

where $\overline{X}^T = \{x_1,x_2,...,x_n\}$ design variables.

The use of the term "minimize" means that any optimization scheme will locate the minimum function value. If a maximum is desired multiplying the objective function by negative one (-1.0) will convert the problem into an equivalent minimization. The two general approaches to solving this problem are direct methods and sequential unconstrained minimization techniques (SUMT) (Figure 2.1).

Direct methods incorporate information about the constraints directly into the optimization problem. Sequential linear programming (SLP) (17) is one such method where the problem and constraints are first linearized by a Taylor series expansion. The resulting linear problem is solved and the process is repeated until the nonlinear minimum is found. Another approach is the method of

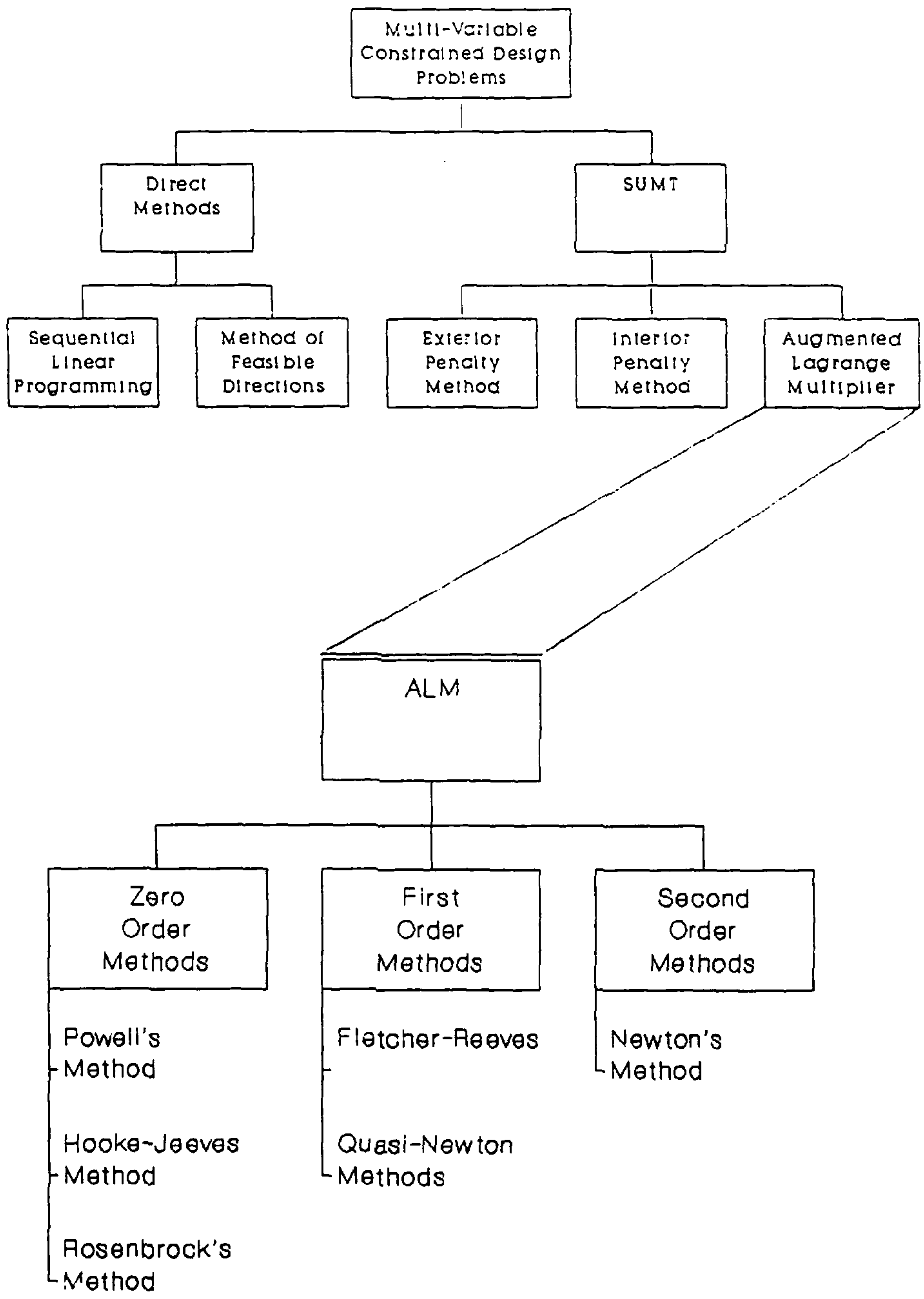


Figure 2.1 Problem Solution Techniques.

feasible directions. From a constraint boundary the gradients of the objective function and any active constraints are determined and a linear approximation of the problem at that point ($\bar{X}$) is found. A search direction is then calculated that reduces the objective function without violating the active constraints. Constraints are active if the design vector is near the constraint boundary so that any small move in that direction will intersect the constraint or produce an infeasible design. It should be noted that all these direct methods require gradient information for the objective and constraint functions.

The second approach, SUMT was first developed by Fiacco and McCormick (12) and incorporates the constraints into a pseudo-objective function that can be minimized by unconstrained techniques. The general form for the SUMT is

$$\Phi(\bar{X},r_p) = F(\bar{X}) + r_p P(\bar{X}). \qquad 2.1$$

Here Φ is the pseudo-objective function, $F(\bar{X})$ the original objective function, r_p a scalar multiplier that determines the magnitude of the penalty, and $P(\bar{X})$ is a penalty function that is determined from the constraints. The penalty function affects $\Phi(\bar{X},r_p)$ only when the corresponding constraint is violated. When the pseudo-objective function is minimized, the original constrained objective function is minimized. Three current methods used to solve Equation 2.1 are the exterior penalty function, the interior penalty function, and the augmented

Lagrange multiplier (ALM).

The exterior penalty function method creates a $P(\bar{X})$ that penalizes the design only when the constraints are violated. The interior penalty function method penalizes the design as it approaches the constraint boundary from within the feasible region. Constraint violations are not allowed and the initial design vector must start in the feasible region of the parametric space (17).

The ALM is a modification of the Lagrange multiplier method for functions with equality constraints. The inequality constraints are modified into equality constraints and incorporated into the Lagrange multiplier equation, therefore the name augmented Lagrange.

For SUMT the problem is converted to a sequence of unconstrained minimizations of n-variables. There are three classes of methods used to solve the unconstrained multivariable minimization problem; zero, first, and second order.

Zero order methods use no explicit derivative information to locate the minimum. These methods are best when derivatives cannot be calculated or are difficult to determine, but they do generally require more function evaluations to obtain convergence. Powell's method of conjugate directions is a widely used method of this class (8). The direction vectors $\bar{S}^i$ and $\bar{S}^j$ are conjugate to each other if

$$(\bar{S}^i)^T \bar{H} \bar{S}^j = 0. \qquad 2.2$$

where $\bar{H}$, the Hessian, is the matrix of second derivatives. Powell's method assumes that a quadratic approximation can be made of the objective function and proceeds to build a corresponding approximation of the Hessian. For a quadratic function, a minimum exists when then Hessian is positive definite. This method has produced many variations, most notably by Brent (4). These subsequent variations attempt to improve determination of the search directions. Rosenbrock's method (5) generates orthogonal search directions to improve convergence. The method of Hooke-Jeeves (8) uses the coordinate unit direction vectors as the search directions and uses an acceleration step during the search. Both Rosenbrock and Hooke-Jeeves utilize the direction of the change in the design vector between complete search sets to accelerate the minimization process.

First order methods rely on computed first derivatives to determine search directions and will converge more quickly than zero order methods for most quadratic functions. The method of Fletcher-Reeves (9) generates gradient-based directions that are conjugate to each other. A class of methods know as quasi-Newton methods approximate the inverse of the Hessian matrix and use this approximation for generating the search directions, without actually requiring second derivative calculations (13).

Davidon-Fletcher-Powell and Broyden-Fletcher-Goldfarb-Shanno are the two the most common quasi-Newton methods, differing only in the way in which the search directions are updated.

Second order methods utilize both first and second derivative information. Newton's method (13) expands the objective function and constraints using a second order Taylor series expansion and solves for the search direction matrix $\bar{S}$ defined by

$$\bar{H}\,\bar{S} = -\nabla\bar{F}. \qquad 2.3$$

If the function is quadratic, the method will converge in one iteration. Both first and second derivatives must be provided.

The interior ballistic model selected (see Section 3) has multiple variables and is constrained. For some propellents there are dependencies that will not allow the model to evaluate certain combinations of parameters, producing holes in the parametric space. The number and combinations of analytical and empirical equations in the model in addition to a different surface area and volume regression equation for each propellent make the evaluation of the first derivatives difficult. These factors make the direct methods, with their dependence on explicit derivative information, less desirable than SUMT using zero order methods.

Of the SUMT methods, the exterior penalty function method approaches the solution from the infeasible region. If this method is terminated early, it may lead to an infeasible design. The interior penalty function method approaches the solution from the feasible region. However, it may have problems dealing with discontinuities of $\Phi(\bar{X}, r_p)$ at the boundaries because of the way the penalty function is generated. The ALM will approach the solution from either the feasible or infeasible region and will ensure constraint compliance at the solution and therefore is the method of choice for this thesis.

The selected interior ballistic model is time efficient. Since the parametric space is nonlinear and this is a first try at this problem, the choice of zero-order methods is indicated. Powell's method provides good convergence and uses the idea of conjugate directions without an explicit dependence on derivative information. Hooke-Jeeves with fixed orthogonal search directions combined with an acceleration step is robust.

Optimization methods are tools for assisting the engineer in design and analysis, not for replacing him. Optimization techniques, when properly applied, result in more efficient and economical designs. A more correct description of the optimization process is "design improvement". Despite the best algorithm and applications, few designs are truly the "best designs". Some advantages from including optimization in design are (17):

1. A reduction in design time, especially when one scheme can be applied to numerous problems.
2. A systematic design procedure.
3. A wide variety of design variables and constraints can be handled.

The following disadvantages are also present:

1. Computational time increases as the number of variables increase. This can make the process prohibitively expensive or numerically ill-conditioned.
2. The process does not have experience to draw on during problem solving.
3. If the analysis is not theoretically precise, the results of the process may be misleading.

2. Ballistics.

Ballistics is the science that deals with the propulsion, flight, and impact of projectiles from guns. Ballistics is organized into three phases. Interior ballistics is the propulsion of the projectile inside the gun system. Exterior ballistics is the flight of the projectile through the atmosphere. Terminal ballistics is the impact and penetration of the projectile into the target. The sequence of events from the ignition of the propellent in the projectile, to departure of the projectile from the stabilizing tube is the interior ballistic cycle and the subject of this thesis.

Ballistics started as an art not a science. Initially interior ballistics was not differentiated from general ballistics because there was no practical way to measure muzzle velocity or pressure in the gun. All that could be said was that given a certain charge mass, projectile, gun, and angle of elevation a certain range could be obtained (7).

Prior to valid theoretical models and the ability to solve them, the practical approach was to solve the problem experimentally. For example, LeDuc (11) fit a hyperbolic curve to experimental data and generated ballistic tables. Some analytical models existed, but their solution was not practical for day to day use. A workable form of the analytical solution did not come until Charbonnier in 1908 (11). Numerous assumptions and simplifications were necessary, since accurate measurement of the pressure-time curve was still not possible.

The development of a reliable piezoelectric gauge around 1935 provided the means to accurately record the pressure-time events in the gun and provided the impetus to connect interior ballistics to the physics and chemistry. The central problem was still the same but more questions could be asked, and answered, by combining theoretical models and empirical data.

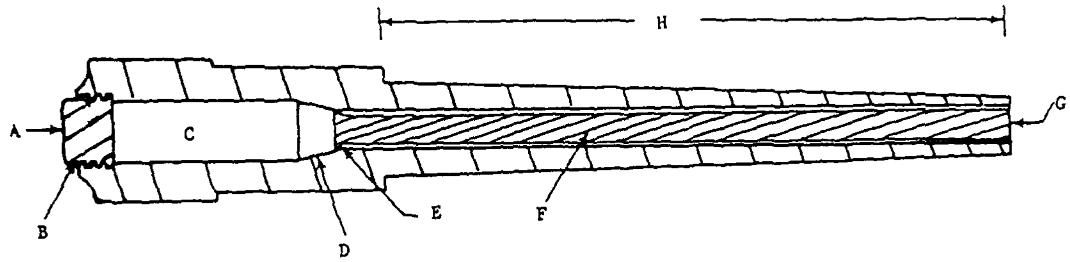
The development of the digital computer caused a change in ballistic modeling. Before the digital computer, closed-form solutions to the governing differential equations or tabular and data curve fitting were predominant. Most notable of the latter were the ballistic tables of Bennet (11) in 1921, some of which are still in use today. The first uses of digital computers were to solve the governing differential equations. In 1962 Baer and Frankle (2) introduced the first direct numerical solution of the ordinary differential equations of interior ballistics on a case by case basis. The solution of the one dimensional (1-D) partial differential equations began in the late 1960's. The first 1-D code was developed by Baer, and subsequently numerous 1-D models have been developed, most notably NOVA (16). The modeling of flame speading phenomena, two phase flow, the condensed propellent and products of combustion are examples of work to improve the simulation of the events in the gun. These are active research efforts(11).

One of the most widely used interior ballistics models today is called IBHVG2 (Interior Ballistics of High Velocity Guns, version 2) (1). It was derived from the Baer-Frankle methodology and includes elements of MPRGUN (Multipurpose Gun Code) (2). It is a lumped parameter model in that it assumes the reaction chamber is well mixed and represented by the rate of burning. The results from IBHVG2 correspond with experimental data and there is a high degree of reliability in the model. Projectile Design and Simulation PRODAS (6) is a model in use that takes projectile design through all three ballistic phases. It uses IBHVG2 as its interior ballistics model. Another lumped parameter model, IBRGAC (Interior Ballistics Model, Robbins-Gough-Anderson, Chambrage) (15), is derived from the NATO technical cooperative program (TTCP), model IBRGA. It is used both as a design tool and to verify predicted results from other codes. Its primary advantages are that it is a straightforward code, not expensive to run, and based on an accepted international model. For these reasons this code is selected as the interior ballistic model to be used in this research.

3. Gun Nomenclature.

The typical gun system consists of a fire control system, a cannon (Figure 2.2), and a round of ammunition (Figure 2.3). The fire control system calculates the exterior ballistic solution for the flight of the projectile. It applies the correction to the elevation and deflection of the gun tube prior to firing.

The cannon is a tube that is closed at one end for firing. The barrel provides a guide and support for the projectile as it is accelerated by the impulse of the propellent gases during the interior ballistic cycle. The breech is opened to allow the projectile to be loaded and is closed for firing. In front of the breech is the



A = breech
B = breechblock
C = reaction (or combustion) chamber
D = shoulder
E = beginning of lands
F = rifled bore
G = muzzle
H = chase

Figure 2.2 Typical Tank Gun Nomenclature (11).

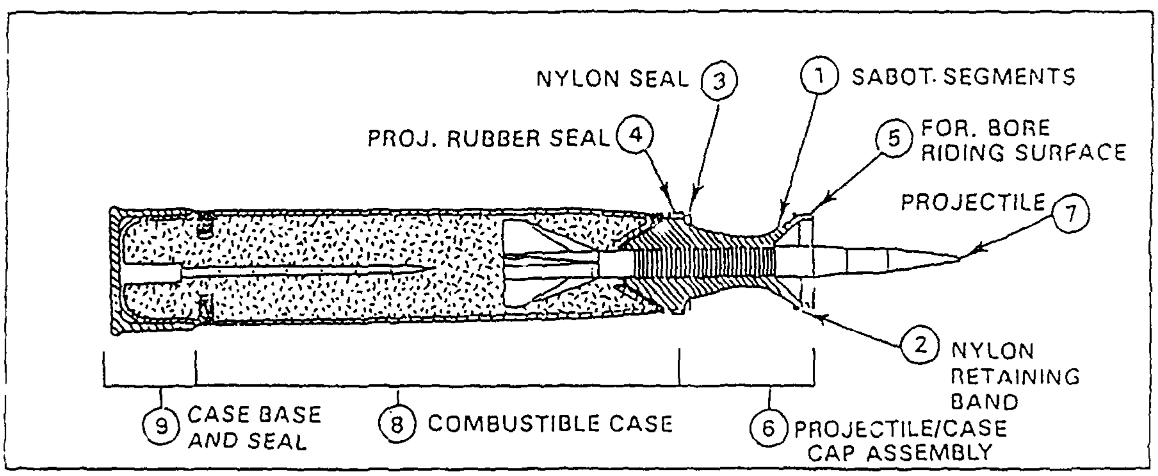


Figure 2.3 Typical Kinetic Energy Tank Round.

reaction chamber that usually has a greater diameter than the remainder of the barrel. It holds the propellent and it is here that ignition and the initial pressure build-up occurs. At the front end of the reaction chamber is an area whose walls taper down to the barrel diameter. This area is called the shoulder. Forward of the shoulder the gun barrel has a uniform diameter, called the gun bore, which continues to the muzzle.

There are two ways to stabilize the projectile in flight. The first way is to impart spin to the projectile as it is traveling down the gun tube. To do this the gun barrel is rifled. Parallel grooves are cut into the barrel that twist down the tube. The rotating band that translates the twist of the rifling to projectile spin is part of the case cap assembly. It is engraved by the rifling as the projectile travels down the tube. The other method is to fin stabilize the projectile. This is done by attaching a boom and fins to the projectile. Normally a smooth bore gun is used and the rotating band seals the propellent gases behind the projectile. In both cases the pressure behind the projectile must overcome the resistance from the rotating band/gun tube interface.

The base of the round is the cartridge case. It holds the propellent and ingiter and is designed to fit snugly in the reaction chamber. The part of the round that travels to the target is either a chemical energy or kinetic energy projectile. The remainder of the round is called the case cap assembly. It consists of the parts necessary to secure and stabilize the projectile in the gun tube. It is discarded by aerodynamic drag after the projectile leaves the gun.

A chemical energy projectile has an explosive charge that detonates upon impact. All the energy needed at impact is provided by this charge. The terminal velocity is not critical. A kinetic energy projectile does not contain any explosive charge. Its destructive force is dependent upon its kinetic energy at impact. The kinetic energy is given as

$$KE = \frac{1}{2}MV^2, \tag{2.4}$$

where M is the mass of the projectile and V is the velocity at impact. It is essential that kinetic energy projectiles have high velocity and mass.

CHAPTER III

OPTIMIZATION METHOD

1. Introduction.

This chapter develops the specific optimization methods and computer code that will be applied in Chapter V to the interior ballistic problem. The code is set up to accept the physical variables for ballistics that are explained in Chapter IV.

2. General Problem Statement.

The nonlinear constrained optimization problem is stated as

Minimize: $F(\overline{X})$...............objective function.

Subject to:

$g_j(\overline{X}) \leq 0$ $j=1,m$..........inequality constraints

$h_k(\overline{X}) = 0$ $k=1,l$..........equality constraints

$x_i^{(lower)} \leq x_i \leq X_i^{(upper)}$ side constraints

$i=1,n$.

Here the design variables are viewed as the vector $\overline{X}$ given as

$$\overline{X}^T = \{x_1, x_2, \ldots, x_n\},$$

where the superscript T means transpose.

3. Augmented Lagrange Multiplier Meth~d.

The material in the background section of Chapter II justified the choice of the augmented Lagrange multiplier

(ALM) method for solving the constrained interior ballistics problem.

The ALM includes all constraint conditions in the optimization scheme. This is done by generating a pseudo-objective function that combines the objective function with the equality and inequality constraints as in Equation 3.1. Side constraints are included in the inequality constraint set. The pseudo-objective function is then minimized as an unconstrained function of the n design variables and (m+1) Lagrange multipliers. Minimizing the new objective function results in the minimum of the original cost function with all constraints satisfied. The form of the general Augmented Lagrangian is (17)

$$A(\overline{X},\lambda,r_p)=F(\overline{X})+\sum_{j=1}^{m}[\lambda_j\psi_j+r_p\psi_j^2]+\sum_{k=1}^{l}\{\lambda_{k+m}h_k(\overline{X})+r_p[h_k(\overline{X})]^2\}, \quad 3.1$$

where

$$\psi_j = \max[g_j(\overline{X}), -\lambda_j/2r_p]. \quad 3.2$$

Here $F(\overline{X})$ is the function to be minimized, $h(\overline{X})$ the equality constraints and $g(\overline{X})$ the inequality constraints. The λ's are the Lagrange multipliers. They are a measure of the magnitude of the constraint violation and are updated between iteration as follows:

$$\lambda_j^{p+1} = \lambda_j^{p} + 2r_p\{\max[g_j(\overline{X}), -\lambda_j^{p}/2r_p]\}, \quad j=1,m \quad 3.3$$

$$\lambda_{k+m}^{p+1} = \lambda_{k+m}^{p} = 2r_p h_k(\overline{X}). \quad k=1,l \quad 3.4$$

The r_p is a scaling factor that weights each constraint. It is updated by a constant multiplying factor

$$r_p = \gamma r_p. \qquad 3.5$$

An upper limit r_{pmax} is also established so that r_p does not increase indefinitely.

Given the initial conditions, Equation 3.1 is minimized. If a solution is found, the algorithm is exited. If not, the λ's and r_p are updated by Equation 3.3, 3.4, and 3.5, and the process continues (Figure 3.1).

The ALM is considered successful when the change in the λ's, the change in the original objective function, and the change in the constraint functions are within specified tolerances between consecutive iterations. $A(\bar{X},\lambda,r_p)$ is minimized by a suitable unconstrained minimization method.

As discussed earlier, the zero order methods selected are Powell's and Hooke-Jeeves. These approaches solve the problem by function evaluations alone and do not use gradient information to locate the minimum.

4. Powell's Method.

Powell's is one of the most popular and reliable of the zero order methods (8). It performs n+1 line searches per iteration. The method assumes quadratic behavior of the function and generates directions that are conjugate to an approximation of the Hessian matrix. The matrix $\bar{H}$ is initialized as an nxn identity matrix. The columns are initially set to have the coordinate directions as the directions. After each set of searches along all current directions, the search directions $\bar{S}^i$ are updated by

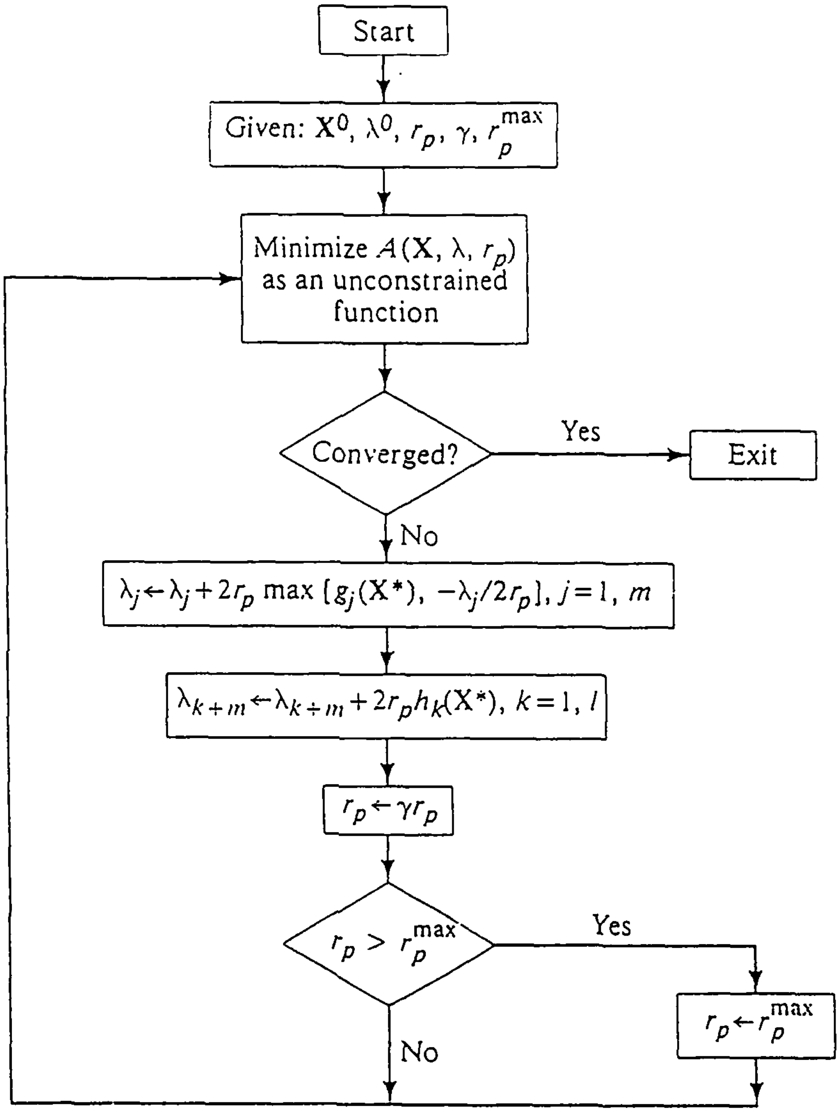


Figure 3.1 ALM Method Algorithm (17).

$$\bar{S}^i = a_i \bar{S}^i, \qquad i=1,n, \qquad 3.6$$

where a_i is the scalar multiplier determining the amount of change in $\bar{X}$ for the ith direction. After n searches, a new direction S^{n+1} is generated by connecting the initial $\bar{X}$ with the current $\bar{X}$ and the n+1 search is performed (17). If the process has not converged the search directions are reassigned as

$$\bar{S}^i = \bar{S}^{i+1}. \qquad i=1,n \qquad 3.7$$

and become the new set of search directions.

In each direction, Powell's method uses a uniform grid search to determine an interval within which a local minimum exists. Either a three point quadratic approximation or the Golden Sections method is used to locate the minimum within the given interval. Although a quadratic approximation is less expensive in terms of function evaluations, it is less robust than the Golden Sections method. The convergence criteria is the change in the design vector, within a specified tolerance, from iteration to iteration.

5. The Hooke-Jeeves Method.

Hooke-Jeeves uses the coordinate directions as exploratory search directions. Hook-Jeeves searches in discrete steps for each direction. After a complete set of exploratory searches, a scalar multiplier accelerates the search in the direction indicated by $\Delta\bar{X}$. If the objective function value is not improved after the acceleration step, the method returns to the previous $\bar{X}$, reduces the search increment and continues. It may require more function evaluations since Hooke-Jeeves always makes use of the coordinate directions, regardless of the behavior of the function. However, it is less likely to fail through numerical ill-conditioning. Convergence is achieved when the search increment is reduced to a value that is less than a predetermined tolerance.

6. Description of the Optimization Code.

The optimization code is written by the author in FORTRAN 77. The code is designed as a series of shells. The outer shell contains the ALM subroutines. In the next shell are the one dimensional unconstrained minimization subroutines. In the inner most shell are the objective and constraint function subroutines. A copy of the code is included in Appendix I. A short description of the code follows.

There is one master header file, 'declarations.ins.f'. This file contains the variable declarations, parameters, and common block definitions for the outer two shells. It is appended to the beginning of each subroutine and includes common and passed variables. Each subroutine declares local variables as needed.

The main program is called 'optimum.ftn'. It controls the process and calls the subroutines that monitor the optimization. All user interaction is done in 'optimum'. From this program the following subroutines are called.

1. read_data: This subroutine reads the interior ballistics code input records and assigns the initial values to the design vector.
2. powell: It drives Powell's method and keeps track of search directions and convergence criteria.
3. hook_jeeves: This is the Hooke-Jeeves algorithm and controls the search and convergence criteria.
4. check_print: This program prints out the current ALM iteration values and other diagnostic information.
5. tol_test: This performs the ALM convergence test of the equality, inequality and original objective function convergence criteria.
6. update: Here the updates of r_p and the λ's for the ALM pseudo-objective function by Equation 3.3, 3.4, 3.5 are performed.
7. printit: This subroutine prints the final objective function value and the final design vector.

The second shell is the unconstrained minimization shell that contains the method of Powell and Hooke-Jeeves. The Hooke-Jeeves subroutine is self contained and calls no other subroutines. The subroutine 'powell' calls the subroutine 'search' and this subroutine calls the following subprograms.

8. ugrid_1d: The uniform grid search algorithm isolates the interval where the minimum is located by conducting a one dimensional unconstrained line search of uniform step sizes until such an interval is found.

9. gold: The Golden Sections interval reducer locates the minimum in the interval found by 'ugrid_1d' through the use of an iterative reduction of the interval. This reduction is done with the use of the Golden Section's ratio of 6.18 and 3.82.

10. quad: This quadratic approximation subroutine uses gaussian elimination to solve the system of equations generated by the interval sent from 'ugrid_1d'. It assumes quadratic behavior in the interval.

Both methods call the subroutine 'funx.ftn'. It performs the transformation of the objective function and the constraint functions to the ALM pseudo-objective function, Equation 3.1. Subroutine 'funx' calls the following two subprograms. The first, 'fun_con.ftn', is a user supplied subroutine that contains all of the equality and inequality constraints and evaluates them for each function call. The second, 'fun_int.ftn' is the modified version of IBRGAC described in Chapter IV and found in

Appendix II.

The user is responsible for providing the input data as required by the interior ballistics code, the constraint subroutines, and the subroutines that allow the transfer of the variables from the optimization code to the interior ballistics code. They are 'var_in.ftn', 'var_out.ftn', and part of 'read_data.ftn'. The user must also specify in 'optimum.ftn' the number of variables, equality constraints, inequality constraints, and tolerances.

CHAPTER IV

INTERIOR BALLISTICS

1. Introduction.

The first part of this chapter presents the conservation equations and empirical relationships used in a lumped parameter model to solve the interior ballistics problem. It is primarily drawn from three sources; the IBRGAC User's Manual (15), the IBVGH2 User's Manual (1), and the derivation by Krier and Adams (11). Many of the equations in these sources are repeated from previous work that will not be cited here. The sequence of events during the interior ballistic cycle are described in Section 2. Section 3 defines the projectile equations of motion as a function of time and their dependence on projectile base pressure. The base pressure's relation to mean pressure in the tube and the rate of propellent gas generation is derived in Section 4. In Section 5, mean gas pressure and its dependence on mean gas temperature and rate of propellent gas generation is derived. Section 6 defines the mean temperature and its relationship to the rate of gas generation and losses to the system, which are detailed in Section 7. Section 8 derives the rate of gas generation and discusses propellent properties. In Section 9 the gun recoil equations of motion are derived. Finally, Section 10 lists the modeling approximations used in the previous sections.

The second part of the chapter describes the IBRGAC code and its organization (Sections 11 and 12).

2. Interior Ballistic Cycle.

The interior ballistic cycle starts at propellent ignition. After the propellent is ignited, pressure and heat rapdily increase inside the chamber from the generation of combustion gases (Figure 4.1). Projectile motion begins after this pressure has overcome the resistance caused by the initiation of engraving the projectile's rotating band by the bore. Pressure increases until the rate of volume increase overcomes the rate of propellent gas generation. Acceleration continues as long as there is a pressure differential across the projectile. The interior ballistic cycle ends when the projectile leaves the gun tube.

3. Projectile Equations of Motion.

A gun is a simple heat engine in which chemical energy of the propellent is transformed into kinetic energy of the projectile and heat (16). Newton's Second Law is

$$a = \frac{F}{m}, \tag{4.1}$$

where at time t projectile acceleration equals the net force generated by propellent combustion divided by effective mass. The projectile maintains a constant in-bore mass. The integral of acceleration with respect to

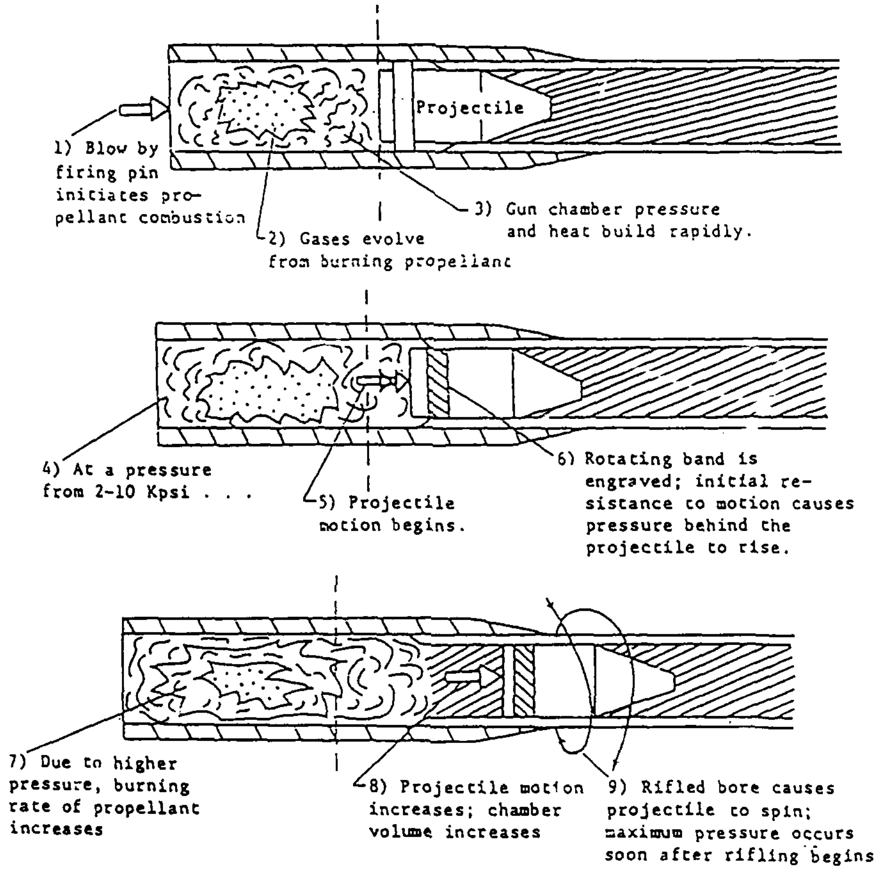


Figure 4.1 Interior Ballistic Cycle (11).

time yields velocity while a subsequent integration gives the distance the projectile has travelled (travel).

Newton's Second Law for this problem is

$$a_p = \frac{A_b}{m_p}(P_b - P_r - P_g), \tag{4.2}$$

where a_p is projectile acceleration, A_b projectile base area, P_b the pressure at the projectile base, P_r the bore resistance to friction and engraving (as an equivalent pressure), P_g the pressure of the air in the tube ahead of the projectile, and m_p the projectile mass. The solution for projectile velocity as a function of time is obtained by determining the pressures in the parenthesis of Equation 4.2 as functions of time and integrating. $P_r(t)$ is interpolated from tabular resistance pressure data and $P_g(t)$ is given as a constant average value. The remaining parameter to be found is the base pressure $P_b(t)$.

Projectile travel, in the ground based coordinate system, is the sum of projectile displacement and the recoiling gun displacement. Summing the integrals with respect to time of projectile velocity and gun recoil velocity (Equation 4.34) gives travel as

$$x = \int_0^t v_p dt + \int_0^t v_{rp} dt. \tag{4.3}$$

4. Base Pressure Derivation.

From initial projectile movement (initial volume is chamber volume minus the volume occupied by propellent) until the projectile leaves the gun tube, the volume occupied by the combustion gases is constantly increasing. As the available volume and quantity of gases increases during propellent consumption, the gases are accelerating down the tube. This is caused by the difference in pressure from the breech to projectile base. A solution for the distribution of pressure, density, and gas velocity in the gun during firing is obtained by the Lagrange pressure gradient approximation. It states that "the velocity of the gas at any instant increases linearly with distance along the bore, from zero at the breech to the full shot velocity at the base of the projectile" (7).

The approximations made in this model are: the chamber is a cylindrical extension of the bore with the same total volume, the entire charge may at any time be treated as gaseous and gas density is uniformly distributed in the gun tube at any time. If the distance from the breech is x and y is the position of the base of the projectile, the gas velocity v_g can be expressed as

$$v_g = \frac{x}{y}\frac{dy}{dt}. \qquad 4.4a$$

The all gaseous propellent charge assumption means that density is

$$\rho = \frac{C_t}{A_b y}, \qquad 4.4b$$

where C_t is the initial mass of propellent and igniter. Uniformly distributed gas density means $\frac{\partial \rho}{\partial x} = 0$. When $x=y$ the gas velocity v_g equals projectile velocity v_p.

Therefore, Newton's First Law can be written as

$$\frac{d^2y}{dt^2} = \frac{A_b}{m_p}(P_b - P_r - P_g), \qquad 4.4c$$

where previous definitions apply. Integrating the equation of motion for the gas using the previous three equations gives the pressure $P(x)$ as a function of distance as

$$P(x) = P_b + \frac{Ct}{2m_p}(1 - \frac{x^2}{y^2})(P_b - P_r - P_g). \qquad 4.4d$$

The mean pressure P_m, between the breech and projectile base, can then be determined as

$$P_m = \frac{1}{y}\int_o^y P dx = P_b + \frac{Ct}{3m_p}(P_b - P_r - P_g). \qquad 4.5$$

Solving for the base pressure P_b gives

$$P_b(t) = [P_m(t) + \frac{C_t(P_r + P_g)}{3m_p}] \;/\; [1 + \frac{Ct}{3m_p}]. \qquad 4.6$$

To determine the base pressure $P_b(t)$, the mean pressure $P_m(t)$ must be determined.

5. Mean Pressure Derivation.

Van der Waals' equation of state is (16)

$$(P + a/V^2)(V - B) = \bar{R}T. \qquad 4.7$$

Here $\bar{R}$ is the universal gas constant, V is the molar volume of gas, the term a/V^2 is the increase in pressure due to intermolecular attractions, and B is the decrease in free volume due to the finite volume of the molecules. This means the available free volume for the gas to move about is less than the free chamber volume. At low pressure and densities these volumes are nearly identical. At high pressure and densities the free volume difference is noticeable.

As temperature and pressure increase in the gun, the effect of intermolecular attractions decreases causing the a/V^2 term in Equation 4.7 to become negligible. Van der Waals' equation is reduced to

$$P(V - B) = \bar{R}T. \quad 4.8$$

To use the preceding equation with the mass and volume of gas, $\bar{R}$ is substituted in Equation 4.8 by the relation

$$\bar{R} = R(Vm/V), \quad 4.9$$

where R is the specific gas constant, m the mass, and V the volume of gas. This form of Van der Waals' equation, after simplification, is known as the Noble-Abel equation

$$P(V - mb) = mRT. \quad 4.10$$

In Equation 4.10 the covolume b is defined as the Van der Waals' constant B divided by the molecular weight of the gas. Using mean values of the pressure P_m and temperature T_m, over the temperature range, Equation 4.10 is

$$P_m(V - \sum_i m_i b_i - m_I b_I) = (\sum_i m_i R_i + m_I R_I)T_m. \quad 4.11$$

The subscript I is the igniter and for multiple propellents the subscript i indicates the ith (i=1,n) propellent. Substituting for the specific gas constant R in the above equation with the propellent force F, defined as

$$F = RT_o, \quad 4.12$$

with T_o being the adiabatic flame temperature of the product gases, the mean pressure P_m can be stated as

$$P_m = T_m[\sum_i \frac{F_i m_i}{T_{oi}} + \frac{F_I m_I}{T_{oI}}]/(V - \sum_i m_i b_i - m_I b_I). \qquad 4.13a$$

At time t, the mean pressure is calculated from Equation 4.13a for values of the mass of gas present, or

$$P_m(t) = T_m(t)[\sum_i \frac{F_i m_i(t)}{T_{oi}} + \frac{F_I m_I}{T_{oI}}]/(V(t) - \sum_i m_i(t) b_i - m_I b_I). \qquad 4.13b$$

The mass of gas present $m_i(t)$ at time t is equal to the fraction of propellent mass burned $z_i(t)$, Equation 4.31, multiplied by the original propellent mass C_i or

$$P_m(t) = T_m(t)[\sum_i \frac{F_i C_i z_i(t)}{T_{oi}} + \frac{F_I m_I}{T_{oI}}]/(V(t) - \sum_i C_i z_i(t) b_i - m_I b_I). \qquad 4.13c$$

The volume available for the gases at time t is

$$V(t) = V_c + A_b x(t) - V_r(t), \qquad 4.14$$

where x(t) is projectile travel, A_b is area of the projectile base, and V_c is initial chamber volume. The total volume of unburnt propellent $V_r(t)$ is calculated from the fraction of mass burned by

$$V_r(t) = \sum_i \frac{Ci}{\rho_i}(1 - z_i(t)). \qquad 4.15$$

The gas density is ρ. To determine the mean pressure $P_m(t)$ in Equation 4.13c, the mean temperature $T_m(t)$ must be known.

6. Mean Temperature Derivation.

From the First Law of Thermodynamics, the energy balance in the gun tube can be stated as: the initial energy of the gases is equal to the internal energy of the gases plus any losses. Losses include work done by and heat transfered from the system and are discussed in the next section. Using average values of specific heats over the temperature range, the initial energy of the gases is

$$E_1 = \sum_i m_i c_{vi} T_{oi} + m_I c_{vI} T_{oI}, \qquad 4.16$$

where m_i is mass, c_{vi} specific heat (at constant volume), and T_{oi} adiabatic flame temperature of the propellent product gases. The same definitions apply for the igniter. The internal energy of the gases in terms of the mean temperature T_m is

$$E_2 = [\sum_i m_i c_{vi} + m_I c_{vI}] T_m. \qquad 4.17$$

Equation 4.17 and 4.18 are used in the energy balance statement. The specific heat c_v is first expressed as

$$c_v = F/[(\gamma - 1)T_o], \qquad 4.18$$

where γ is the ratio of specific heats, and the results are solved for mean temperature T_m to obtain

$$T_m = \frac{[\sum_i \frac{F_i m_i}{(\gamma_i - 1)} + \frac{F_I m_I}{(\gamma_I - 1)} - L]}{[\sum_i \frac{F_i m_i}{(\gamma_i - 1)} T_{oi} + \frac{F_I m_I}{(\gamma_I - 1)} T_{oI}]}. \qquad 4.19a$$

At time t, the mean temperature is calculated from Equation 4.19a as

$$T_m(t) = \frac{[\sum_i \frac{F_i m_i(t)}{(\gamma_i - 1)} + \frac{F_I m_I}{(\gamma_I - 1)} - L(t)]}{[\sum_i \frac{F_i m_i(t)}{(\gamma_i - 1)} T_{oi} + \frac{F_I m_I}{(\gamma_I - 1)} T_{oI}]}. \qquad 4.19b$$

Again $m_i(t)$, the mass of the gas present at time t, is calculated from the fraction of propellent mass burned $z_i(t)$, Equation 4.31, multiplied by the original propellent mass C_i or

$$T_m(t) = \frac{\left[\sum_i \frac{F_i C_i z_i(t)}{(\gamma_i - 1)} + \frac{F_I m_I}{(\gamma_I - 1)} - L(t) \right]}{\left[\sum_i \frac{F_i C_i z_i(t)}{(\gamma_i - 1)} T_{oi} + \frac{F_I m_I}{(\gamma_I - 1)} T_{oI} \right]}. \qquad 4.19c$$

For both the mean pressure and temperature, the derivation of the fraction of mass burned $z_i(t)$ is in Section 8.

7. Work and Losses.

In the previous section there is loss L(t) due to work performed and heat transferred from the system. There are three general classes of loss. The first is work done to the projectile and gun. They are; loss to projectile translation and rotation and recoil of the gun. The second is work lost to the propellents and resistances. They are; energy losses to propellent gas and unburned propellent motion, bore resistance due to engraving and friction and air resistance in front of the projectile. The third is heat transfer to the chamber wall. All these can be written as follows:

$$L(t) = E_{pt} + E_{pr} + E_{rp} + E_p + E_{br} + E_c + E_h. \qquad 4.20$$

These losses and the relevant equations are listed in Table 4.1.

Type of Energy Loss	Equation
projectile translation	$E_{pt} = \frac{1}{2} m_p V_p^2$
projectile rotation	$E_{pr} = \frac{\pi}{4} m_p V_p^2 Tw^2$
recoil of the gun	$E_{rp} = \frac{1}{2} m_{rp} V_{rp}^2$
propellent gas and unburnt propellent *motion*	$E_p = \frac{1}{2} A_b \int_o^y V_g^2 dx = \frac{1}{6} C_t V_p^2$
bore resistance due to engraving and friction	$E_{br} = A_b \int_o^t P_r V_p dt$
loss to air resistance	$E_c = A_b \int_o^t P_g V_p dt$
heat transfer to the chamber walls and gun barrel	$E_h = \int_o^t \dot{Q} dt$

Table 4.1 Table of losses for Equation 4.20.

For energy loss to projectile rotation E_{pr}, Tw is the twist of rifling in turns per caliber. More precisely, Tw is the ratio of complete revolutions to bore diameter for the rifled grooves down the length of the gun tube.

Energy due to heat transfer to the internal chamber walls and gun barrel by convection E_h is assumed to be proportional to the difference of the mean temperature of the system and average temperature of the wall. At time t, this heat loss can be stated as

$$E_h = \int_o^t \dot{Q} dt, \qquad 4.21$$

where

$$\dot{Q}(t) = A_w(t) h(T_m(t) - T_w(t)), \qquad 4.22$$

and $T_w(t)$ is the temperature of the chamber wall. The exposed chamber wall area A_w is

$$A_w(t) = \frac{Vo}{A_b} \pi D_b + 2A_b + \pi D_b x(t), \qquad 4.23$$

where D_b is bore diameter, V_o initial chamber volume, and the heat transfer coefficient is given by

$$h = \lambda \bar{c}_p \bar{\rho} \bar{v}_g + h_o. \qquad 4.24$$

Here $\bar{c_p}$, $\bar{\rho}$, and $\bar{v}_g$ are mean values for the previously defined symbols and h_o is a natural convective term which allows heat transfer if the projectile is not moving. The Nordheim friction factor λ is empirically derived from gun tube experimentation and found to be

$$\lambda = [13.2 + 4\log 10[100.0 D_b]]^{-2}. \qquad 4.25$$

The chamber wall temperature T_w in Equation 4.22 is derived from an energy balance that says

$$\text{heat transfer} + \text{work} = \Delta \text{ internal energy.}$$

The heat transfer is E_h and work is given by E_{br} multiplied by an empirical factor f, the fraction of work done against bore friction that preheats the chamber. The change in internal energy of the chamber wall is

$$\Delta = T_w c_{pw} m_w - T_c c_{pw} m_w, \qquad 4.26$$

where the w subscript indicates the chamber wall properties for specific heat (at constant pressure) and mass of the chamber wall. The initial chamber wall temperature is T_c. Placing the above terms into the conservation equation and substituting mass with density and volume yields

$$E_h + fE_{br} = c_{pw} \rho_w A_w D_w (T_w - T_c). \qquad 4.27$$

Here D_w is the chamber wall thickness. In terms of the chamber wall temperature T_w, Equation 4.27 is

$$T_w = \frac{Eh + fEbr}{c_{pw} \rho_w A_w D_w} + T_c, \qquad 4.28$$

and can be placed into Equation 4.22 for $T_w(t)$ at time t.

8. Propellent and Rate of Burning.

Propellents are composed of compounds that ignite and burn quickly, producing large quantities of gas rapidly. Conventional propellents are primarily nitrocellulose and their basic geometric unit is the grain.

The burning rate of the propellent is the rate at which the surface of the propellent regresses. The empirical equation is the steady state burning law (7)

$$r = \beta P_m^{\alpha}, \qquad 4.29$$

where β is the burning rate coefficient and α is the burning rate exponent. The unit of r is meter per second. Experimental burning rate data are fitted to the equation to determine the coefficients, which are functions of propellent temperature.

The burning of the propellent grains produces the pressures necessary to overcome the initial resistive forces and accelerate the projectile down the gun tube. The model states that grains burn uniformly and without deformation. At constant pressure the mass of the combustion gases produced is proportional to the surface area exposed. As burning continues, the rate of mass produced depends on the surface area as a function of time. The equations for the recession of the exposed surface area for propellents are determined from geometric analysis of the grain.

The mass fraction burning rate $\dot{z}$ is the rate at which the propellent mass is being consumed and therefore the rate gas is generated. The relationship is

$$\dot{z}_i = S_i r_i / V_{gi}, \qquad 4.30$$

where S_i is the remaining propellent grain surface area, r_i is the linear burning rate from Equation 4.29 and V_{gi} is the initial grain volume. Integrating equation 4.30 yields the fraction of mass burned at time t as

$$z_i = \int_0^t \dot{z}_i dt. \qquad 4.31$$

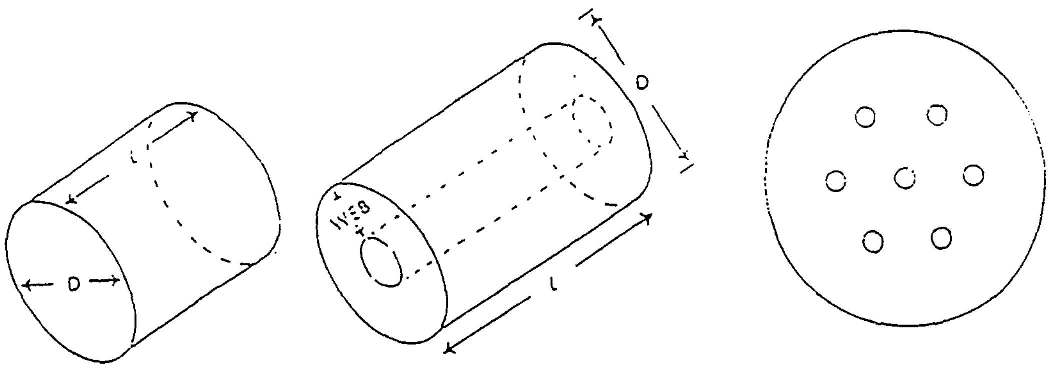
The physical configuration of the grains within the chamber (packing) has an effect on the rate of burning and the resultant chamber pressure. This effect is not considered within the scope of this research.

There are three types of propellent grain geometry: regressive, neutral, and progressive. Regressive burning grains reduce their surface area during burning. Neutral burning grains maintain a constant surface area until consumed. Progressive burning grains increase in surface area as they burn. For progressive burning grains, once the grain has burned a certain distance the perforations intrude upon each other and burning becomes regressive. An example of each type is depicted in Figure 4.2.

Another factor in propellent performance is the density of loading. It is the weight of the propellent in the chamber, divided by the volume of the chamber available to the propellent. The loading density is a measure of how much propellent is present and therefore how many moles of gas will be generated. An increase in the density of loading will generally increase pressure in the chamber.

9. Gun Recoil.

Recoil is the rearward movement of the gun in the ground reference frame during firing. Recoil is caused by reaction to the forward motion of the projectile and propellent gases. Recoil systems are designed to absorb this energy so that the gun will remain stable during



Cord
(Regressive)

Single Perforation
(Neutral)

Seven Perforation
(Progressive)

Figure 4.2 Typical Propellent Grain Burning Types.

firing. The equation of motion from Newton's Second Law is

$$a_{rp} = \frac{Ab}{m_{rp}}(P_{br} - \frac{RP}{A_b} - P_r). \qquad 4.33$$

Here P_{br} is the breech pressure, RP is the resistive force to recoil motion, and the subscript rp means recoiling parts. The acceleration of the recoiling parts is zero until the pressure at the breech is greater than the combined resistive forces to recoil motion and barrel resistance. Integration of a_{rp} gives the corresponding velocity v_{rp} as

$$v_{rp} = \int_0^t a_{rp} dt, \qquad 4.34$$

10. Modeling Approximations.

Modeling the interior ballistic cycle uses the following conditions:

1. The propellent gas mixture is described by the Noble-Abel equation of state. This means that the gases are well mixed and no solid or liquid phases exists.
2. The propellent gas flow is taken to be one-dimensional, inviscid, and compressible.
3. The steady state burning rate law can be used to describe the recession of the rate of the propellent grains.
4. The base of the projectile is flat and perpendicular to the direction of travel.
5. Propellent grains are all the same size and

configuration for a given propellent load. For perforated propellents, all holes are placed in the grain symmetrically.

6. All propellent is ignited simultaneously and uniformly. The igniter is consumed by t=0.
7. All exposed burning surfaces recede at the same rate and perpendicular to the surface. That is, the grains shrink uniformly without deformation.
8. Decomposition of a unit mass of propellent will always liberate the same amount of energy, which heats product gases to the same temperatures.
9. The main constituents of the propellent gas mixture do not suffer further chemical or physical reactions.

11. Description of IBRGAC.

IBRGAC is a lumped parameter interior ballistics code written in FORTRAN. It was developed in 1987 at the Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland and validated by experimental data.

IBRGAC uses the Lagrange and chambrage method for the breech to base pressure gradient. The chambrage gradient equation takes into account the narrowing of the front of the chamber to calculate the pressure gradient in the gun tube. It is demonstrated in the User's Manual that both methods result in comparable projectile performance predictions. The Lagrange method is used exclusively in this research for consistency.

The user provides a input file organized into 9 records that are defined as follows.

1. Record 1; Gun system data and pressure gradient calculation selection flag.
2. Record 1a; If the chambrage gradient is selected, this record is read and contains chamber dimension data.
3. Record 2; Projectile mass, air resistance flag, and *f*, the fraction of work done against bore friction that preheats the chamber, data.
4. Record 3; Barrel resistance point data.
5. Record 4; Recoil data.
6. Record 5; Heat transfer data.
7. Record 6; Igniter data.
8. Record 7; Propellent data (up to 10 propellents).
9. Record 8; Propellent burning rate point data (for each propellent used).
10. Record 9; Time increment data.

The data are read from the input file and printed in the output file. All input data are required to be in the MKS system. The example problem input files are in Appendix III.

The model uses 4th order Runge-Kutta integration to calculate projectile velocity and travel, projectile resistance energy, system heat loss, recoil velocity and travel, and energy loss to air resistance. The time rate of change of mass and surface area of propellent are determined from the linear burning rate.

The algorithm continues until either the projectile has left the tube or the stop time has been reached. The program will terminate for detected errors in input and output records or unacceptable grain dimensions. If time expires before the projectile has exited the tube, current projectile velocity rather than muzzle velocity is displayed. For each time step, elapsed time, acceleration, velocity, travel, breech and mean and base pressures are calculated. Once the program is complete, initial and residual propellent gas energy and all losses from Section 7 are calculated for the cycle.

12. Program Organization.

The code has been reorganized into a main program and six subprograms. This was done to allow integration with the optimization code. A complete listing of each file is in Appendix II. There is one master header file 'intball.ins.f'. This file possesses the variable declarations, parameters and common block definitions. It is appended to each subroutine. Each subroutine declares local variables. A description of each file follows:

1. fun_int; This is the main program and performs the interior ballistic calculations with the exception

of surface area and volume rate of change calculations done in 'prf017' and the loading density in 'mass_check'. It is called by subprogram 'funx' from the optimization program (see Chapter III). All subroutines are called by 'fun_int' except 'read_data'.

2. read_data; This subroutine is called by the optimization code. The input file values are assigned to a backup set of variables and the initial design vector is created.

3. reset_data; The subroutine resets all local variables used in 'fun_int' from their values and sets the working variables to their initial values for each iteration.

4. var_in; This subprogram is modified by the user and assigns the values from the design vector $\bar{X}$ to the respective working variables in 'fun_int'.

5. mass_check; This subroutine determines the density of loading and maximum propellent charge for each problem. The actual propellent volume is calculated and compared to the maximum charge volume. It can be quickly modified to accept any factor for maximum propellent load.

6. prf017; This subprogram determines the acceptability of the propellent dimensions and calculates the mass fraction and surface fraction

of propellent burned.

7. var_out; This subroutine returns the respective working variables to the design vector after each iteration.

CHAPTER V

EXAMPLE PROBLEMS

1. Introduction.

The purpose of this chapter is to specify the equipment used in the example problems, state the problems solved, and critique the results. Section 2 describes the hardware used in the example problems and includes the specifications of the gun, projectile, and propellent. In Section 3 the optimization objective function for the interior ballistics problem is stated, while Section 4 develops the constraints by category. Section 5 explains the parameters initialized in the optimization scheme. The selected example problems, their purpose, organization, input, and output are in Section 6. The analysis of the results is in Section 7.

2. Baseline Equipment.

The gun, projectile, and propellents are described in this section. The gun system is representative of the current tank main gun.

The cannon is 4.57 meters long with a 120 mm bore diameter. The chamber is 54.0 cm long and 15.4 cm in diameter. In the forward 8.0 cm of the chamber its diameter constricts to 12.7 cm, then reduces to 12.0 cm at the barrel. The chamber volume is 9832 cm^3. The gun is smooth bore with no twist.

The design factor of safety n for gun tube strength is 1.15 (19). The yield point strengths σ_{yp} of the gun, as a function of distance down the bore, are:

σ_{yp} (MPa)	Bore Location (m)
696.0	$0.00 \le x_p \le 1.50$
276.0	$x_p = 4.00$
171.0	$x_p = 4.57$

Here x_p indicates the projectile's location in the gun tube. The von Mises-Hencky failure criteria (18) is used to determine the maximum pressure P_{max} for the gun as

$$P_{max} = \sigma_{yp}/1.732*n. \qquad 5.1$$

From 1.50 meters to the muzzle, a 1st order least squares fit provides the distance-pressure functions (P_{max} in MPa, x_p in meters)

$$P_{max}(x_p) = 346.0, \qquad 0.00 \le x_p \le 1.50 \qquad 5.2$$

$$P_{max}(x_p) = 479.8 - 83.20*x_p, \qquad 1.50 < x_p \le 4.00 \qquad 5.3$$

$$P_{max}(x_p) = 502.9 - 91.23*x_p. \qquad 4.00 < x_p \le 4.57 \qquad 5.4$$

The pressure P_{max} is used as the upper limit on breech and base pressure. Breech pressure is checked against Equation 5.2. This pressure occurs at $x_p = 0.0$ for the entire cycle. Base pressure is checked against all three equations as a function of x_p. This pressure occurs at x_p throughout the cycle.

The kinetic energy projectile weighs 9.796 kg, including the case cap assembly. It is fin stabilized and does not require applied spin. The projectile base is assumed to be a flat disk perpendicular to direction of travel.

There are two propellent compounds and three propellent grain geometries. The two compounds resemble the M6 and M8 military propellents. Their thermodynamic properties are listed in Table 5.1. All three grain geometries are cylindrical (cord) propellents with zero, one, and seven perforations. Figure 5.1 shows their critical dimensions. The seven perforation propellent must have the outer perforations (p_o) equally spaced about the center. The three webs (w, w_i, w_o) need not be equal and are determined from the input dimensions L, D, p_i, p_o, and d.

3. Problem Objective Function.

The objective is to maximize projectile velocity for the given conditions, without violating constraints. The objective function is the projectile velocity equation (the time integral of Equation 4.2), including all required ancillary equations discussed in Chapter IV. All function evaluations will be multiplied by negative one (-1.0) to make the objective (maximum velocity) and formulation (function minimization) compatible.

4. Problem Constraints.

The constraints fall into three categories; dimension,

Propellent	Impetus	Adiabatic Flame Temperature	Covolume	Density	Ratio of Specific Heats
	J/g	^{o}K	cm^3/g	g/cm^3	none
Sample (M6)	1135.99	3141	.9755	1.6605	1.23
M8	1168.90	3768	.9550	1.2119	1.62

Table 5.1 Propellent Thermodynamic Properties (15 and 16).

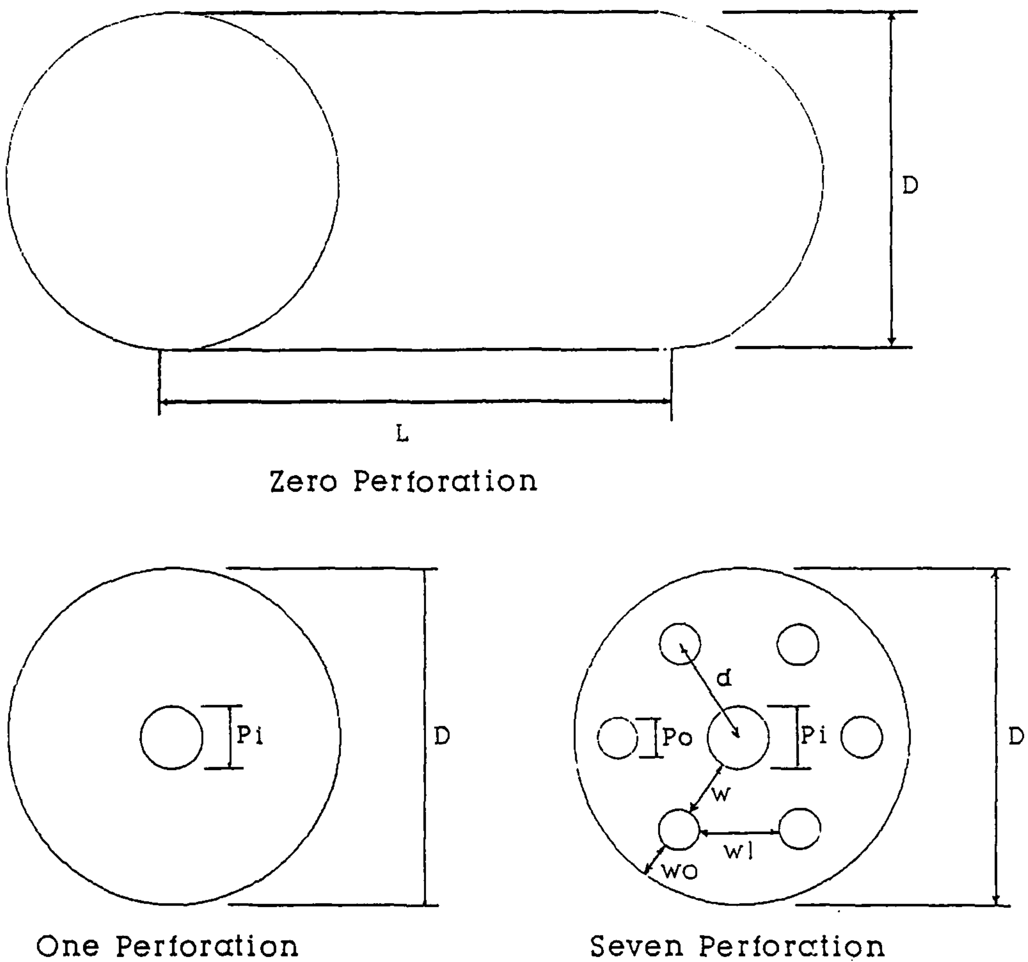


Figure 5.1 Example Propellent Geometries.

mass, and pressure. Constraints include both general and specific criteria and need not directly involve design variables. The constraints are developed for each component of the problem, then as each example problem is developed a specific constraint set is developed from this general set.

The size constraints for the propellent grains include non-negative values for all sizes and mass, in the form

$$x_1 \geq 0,$$

or

$$g_1(\bar{X}) = -x_1. \qquad 5.5$$

For the given chamber size a practical limit of 6.0 cm is put on grain length. This allows propellent grain to lay in the reaction chamber and extend no farther than from chamber wall to ingiter probe in the center of the chamber. The form of this constraint is

$$L \leq 6.0,$$

or

$$g_2(\bar{X}) = \frac{L}{.06} - 1.0. \qquad 5.6$$

Equation 5.6 is normalized so the constraint magnitudes will be comparable to each other. After normalization, the constraints are multiplied by scaling factors to ensure that they are of sufficient magnitude to affect the objective function. For the zero perforation propellent (see Figure 5.1 for propellent dimensions) the constraints are:

1) 3 non-negative constraints (L, D, mass).

2) $L \geq D$.

The one perforation propellent diameter D must be greater that the diameter of the inner perforation p_i, giving

1) 4 non-negative constraints (L, D, p_i, mass),

2) $L \geq D$, and

3) $D > p_i$.

For the seven perforation propellent there are three other size constraints. The first is that the propellent diameter must be greater that the sum of the inner p_i and two outer p_o perforations. The second is that the distance between the perforation centers d must be greater than the sum of the inner and outer perforation radius. Finally, two adjacent outer perforations and the inner perforation form an equilateral triangle. The outer perforation p_o must be less than one side of the triangle d. This leads to:

1) 6 non-negative constraints (L, D, p_o, p_i, d, mass),

2) $L \geq D$,

3) $D > p_i + 2*p_o$,

4) $d > p_i/2 + p_o/2$,

5) $p_o > d$.

The mass constraint for propellents says that propellent volume must be less than chamber volume. This is determined by comparing the total volume occupied for all propellents to initial chamber volume V_c or

$$\sum_i \frac{m_i}{\rho_i} \leq V_c. \qquad 5.7$$

No reduction factor is included to account for volume lost due to packing. Equation 5.7 is the only non-negative mass related constraint, regardless of the number of propellents.

Maximum pressure constraints are both equality and inequality constraints. The maximum pressure in the gun will occur at the breech. This pressure is key to projectile performance. To keep pressure at maximum without exceeding P_{max} the breech pressure is constrained to equal P_{max} so that

$$P_{br} = P_{max}, \qquad 5.8$$

or

$$h_1(\bar{X}) = \frac{P_{max}}{P_{br}} - 1.0. \qquad 5.9$$

The constraint is normalized. The base pressure constraint is an inequality constraint utilizing Equation 5.2, 5.3, and 5.4. In this form

$$485.3 - 86.8*x_p \geq P_b(\text{MPa}), \qquad 5.10$$

where x_p is checked to determine which equation to use.

5. Optimization Initialization.

The values used for the tolerances and multipliers that are required for the optimization method are developed in this section. The multiplier r_p is initialized at 100.0. This is to ensure the constraints affect the objective function value as quickly as possible. The r_p update factor γ is set at 2 to provide a geometric increase for the magnitude of r_p per iteration. The maximum multiplier value r_{pmax} is set at 10^8 to provide a reasonable upper limit for r_p. All Lagrange multipliers λ's are initially set to 1.0 to provide a neutral start point. The update formulas will determine final λ values.

From Chapter II the convergence criteria for the ALM is set at .2. This is appropriate for the magnitude of the velocity (10^4) and the λ's of the scaled and normalized constraint values. The tolerance sent to the line searches is .0001. This ensures that the search is not more precise than the grain manufacturing tolerance, .007cm (19). This value is not inconsistent since all input values are converted to meters in the interior ballistics code so that .0001 is equivalent to a .01 tolerance for centimeter values.

6. Example Problems.

The optimized and automated design process proposed must be able to accomplish several tasks. This process must attain, at least, the performance level of current design methods. It must achieve a practical optimum design regardless of the relative size of the parametric space. It also must be flexible, work for various combinations of variables, and be easy to use. The example problems are designed to answer these questions.

In Appendix D of the IBRGAC User's Manual there is a seven perforation propellent "optimized" under current

design techniques for a maximum breech pressure of 346 MPa. This design was performed by holding all variables constant and varying only the propellent inner web. The model was executed once for each each web increment and the mass was incremented manually until 346 MPa was attained.

The first example problem addresses the questions of whether the proposed method can attain comparable projectile performance compared to current design methods and does the current method attain an optimum design? The optimization process is started from the current best design for the seven perforation propellent and from a random point in the parametric space. These results will indicate the optimization process performance against the current design procedures and provide a measure of the ability of the process to search the same parametric space from different points to achieve an optimum.

The second example problem examines whether the process continues to work for a slightly more complex parametric space. Two propellents of differing geometries are used i.e. one and seven perforation. This example problem has two parts. In Part 1 an optimum design is determined from the initial propellent geometry. In Part 2 the optimization is restarted from the final design of Part 1 to determine if the process has converged at the optimum design in the parametric space.

The third example problem again addresses the question of the optimization process performance in a larger parametric space. Three propellents are used; zero, one, and seven perforation.

The fourth example continues to examine the optimization process performance, in a different parametric space. In this example, two seven-perforation propellents with different thermodynamic properties are used. As in Example 2 a second optimization iteration is performed starting from the first iteration's final design. This checks the optimization performance of a different point in a different space than Example 2.

All four examples will address the question of the optimization method's flexibility and ease of use. The first three will also allow a comparison of the effects of a gradually increasing parametric space on the optimization process and projectile performance.

The input files for each example problem and a sample output file are listed in Appendix III. The output file demonstrates the format and calculated results available from IBRGAC.

Each example is organized into the following parts: problem statement, initial and final design table, initial and final performance table, constraint set, ALM iteration history, pressure-time, and pressure-travel charts.

The problem statement covers the conditions of the problem and its objective. Also in the problem statement is the identification of figure and table numbers appropriate to the problem and the specification of the design vector $\bar{X}$ for the example. Numerically subscripted components of the design vector indicate propellent type in multiple propellent examples. The number of constraints for the problem is also stated.

The initial and final design table shows the original specifications for the propellents used for the example and compares them to the final values obtained by the optimization process. The initial and final performance table compares the velocity and maximum pressures of the initial and final designs. This allows a comparison of the change in performance resulting from the optimization.

The constraint set is taken directly from the subroutine used in the code. This allows all of the constraints in their actual format to be examined. An explanation of each constraint is included in the subroutine comments. The constraint set is generated from Section 4 derivations for different propellent types. All example problems have the pressure (Equation 5.9 and 5.10) and mass (Equation 5.7) constraints included.

The ALM history shows the performance of the optimization method for each example. The number of ALM iterations, the number of function calls, the objective function value (FCOST), and the Lagrange pseudo-objective function value (ALM) are measures of the optimization method's performance. This allows a comparison of the relative performance of the method in various parametric spaces.

The pressure-time and pressure-travel profiles for the initial and final designs allows a graphic portrayal of the performance of the propellent design and a time history of the projectile's velocity. The pressure-time curve indicates the impulse for the propellent while the pressure-travel curves gives the work performed. An increase in the area under either curve indicates an increase in projectile velocity.

The last figure of each example is a breech pressure comparison graph. It show the difference between the initial and final breech pressure as pressure-travel profiles. This graphically displays the change in work performed on the projectile from initial to final design.

Each example problem and its results are listed in order and are analyzed in Section 7.

Example 1: This example contains two parts. First, the process is started from the "optimized" design from Appendix D of the IBRGAC User's Manual. This is done to check the performance of the optimization scheme against the current design method, as described earlier in this section. Second, the same propellent is used but started at a different point. This is done to compare the optimum design attained from two distinct starting points in the same parametric space. The initial and final design results are given in Table 5.1.1. The initial and final performance values of the optimization are in Table 5.1.2. The set of constraints are stated in Table 5.1.3. The ALM iteration history for Example 1a is given in Figure 5.1.1. The pressure-time and pressure-travel profiles for Example 1a are Figures 5.1.2 and 5.1.3. The pressure differential curve is Figure 5.1.4.

The parametric space for this problem consists of six variables, the five critical dimensions for the seven perforation propellent used and its mass. The design vector $\overline{X}$ for Example 1 is

$$
\begin{aligned}
x_1 &= L, \\
x_2 &= P_i, \\
x_3 &= P_o, \\
x_4 &= D, \\
x_5 &= d, \\
x_6 &= \text{mass}.
\end{aligned}
$$

The propellent dimension terms are defined at the beginning of the thesis. There is one equality constraint and 13 inequality constraints.

Initial Values	Example 1a Propellent 1	Example 1b Propellent 1
Type	Sample	Sample
No. Perf	7	7
Mass (kg)	8.70	8.90
Dimensions (cm)		
L	3.175	4.000
D	1.702	2.000
p_i	.0508	.0200
p_o	.0508	.0400
d	.2807	.4000

Final Values	Example 1a Propellent 1	Example 1b Propellent 1
Type	Sample	Sample
No. Perf	7	7
Mass (kg)	8.91	8.94
Dimensions (cm)		
L	3.225	4.370
D	.987	1.040
p_i	.0108	.0100
p_o	.0208	.0400
d	.2507	.2700

Table 5.1.1 Initial/Final Propellent Values.

Example 1a

	Initial	Final
Projectile Velocity (m/s)	1398	1408
Max Breech Pressure (MPa)	346	345
Max Base Pressure (MPa)	240	237

Example 1b

	Initial	Final
Projectile Velocity (m/s)	573	1407
Max Breech Pressure (MPa)	64	345
Max Base Pressure (MPa)	44	237

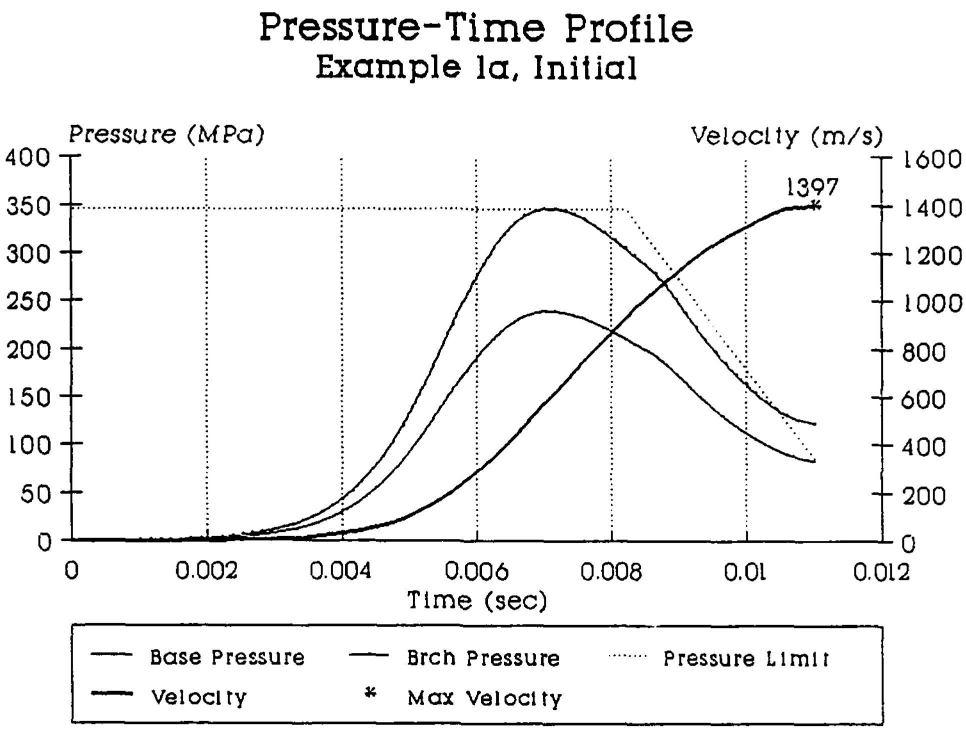
Table 5.1.2 Initial/Final Performance Values.

```
C***********************************************************************
      SUBROUTINE FUN_CON7(X,NOVAR)
C***********************************************************************
C  THIS IS THE CONSTRAINT SET FOR EXAMPLE 1. 7 PERFORATION GRAIN.
C         FH = 1
C         FG = 13
C.......................................................................
%INCLUDE 'declarations.ins.f'

      COMMON/limits/dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,total_vol_prop,
     +       cham_vol
      REAL*4 X(NOVAR),dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,pmax,cham_vol,
     +       total_vol_prop

C  FOR 7 PERF PROPELLENT..................................dimensions..
c  fh1  :max breech pressure constraint                      Pa
c  fg1  :prop grain length .GT. 0 constraint                 m
c  fg2  :inner perf diam .GT. 0 constraint                   m
c  fg3  :outer perf diam .GT. 0 constraint                   m
c  fg4  :prop grain diam .GT. 0 constraint                   m
c  fg5  :dist between perf centers .GT. 0 constraint         m
c  fg6  :mass .GT. than 0 constraint                         kg
c  fg7  :prop diam .GT. (inner+outer perf diams) constraint
c  fg8  :dist between perf centers .GT. (inner + outer radius) constraint
c  fg9  :length .GT. diameter constraint
c  fg10 :max length for the cord 6cm
c  fg11 :equilateral triangle requirement
c  fg12 :max base pressure constraint
c  fg13 :maximum volume of propellent cannot exceed the space in the chamber

      FG(1) =  1000*(-x(1))
      FG(2) =  1000*(-x(2))
      FG(3) =  1000*(-x(3))
      FG(4) =  1000*(-x(4))
      FG(5) =  1000*(-x(5))
      FG(6) =   100*(-x(6))
      FG(7) =   100*(2*x(3)/x(4) + x(2)/x(4) - 1.0)
      FG(8) =   100*(.5*x(3)/x(5) + .5*x(2)/x(5) - 1.0)
      FG(9) =   100*(x(4)/x(1) - 1.0)
      FG(10)=   100*(x(1)/.06 - 1.0)
      FG(11)=   100*(x(5)/x(3) - 1.0)

      pmax  = 3.46e8
      FH(1) =  pmaxbr/pmax - 1.0

c  Determine acceptable pressure ........................................
      if (dpmaxba.gt.1.5.and.dpmaxba.le.4.0) then
        pmax = -8.320e7*dpmaxba + 4.708e8
      else if (dpmaxba.gt.4.0.and.dpmaxba.le.4.57) then
        pmax = -9.123e7*dpmaxba + 5.029e8
      else if (dpmaxba.gt.4.57) then
        pmax = .86e8
      end if
      FG(12)= 5*(pmaxba/pmax - 1.0)
      FG(13)= 100*(total_vol_prop/cham_vol - 1.0)

      RETURN
      END
C  END OF FUN_CON.........................................................
```

Table 5.1.3 Example 1 Constraint Set.

```
NUMBER OF DESIGN VARIABLES       : 6
NUMBER OF EQUALITY CONSTRAINTS   : 1
NUMBER OF INEQUALITY CONSTRAINTS : 13

YOU HAVE SELECTED POWELLS METHOD

AT ITERATION NUMBER 1 AND CALL NUMBER 48
CURRENT  FCOST =  -1545.046
CURRENT  ALM   =  -1499.489

AT ITERATION NUMBER 2 AND CALL NUMBER 79
CURRENT  FCOST =  -1501.847
CURRENT  ALM   =  -1451.523

AT ITERATION NUMBER 3 AND CALL NUMBER 110
CURRENT  FCOST =  -1450.726
CURRENT  ALM   =  -1409.354

AT ITERATION NUMBER 4 AND CALL NUMBER 162
CURRENT  FCOST =  -1394.435
CURRENT  ALM   =  -1401.103

AT ITERATION NUMBER 5 AND CALL NUMBER 213
CURRENT  FCOST =  -1415.718
CURRENT  ALM   =  -1405.178

AT ITERATION NUMBER 6 AND CALL NUMBER 248
CURRENT  FCOST =  -1408.722
CURRENT  ALM   =  -1405.803

AT ITERATION NUMBER 7 AND CALL NUMBER 300
CURRENT  FCOST =  -1409.944
CURRENT  ALM   =  -1407.968

AT ITERATION NUMBER 8 AND CALL NUMBER 335
CURRENT  FCOST =  -1404.827
CURRENT  ALM   =  -1407.929

AT ITERATION NUMBER 9 AND CALL NUMBER 389
CURRENT  FCOST =  -1409.682
CURRENT  ALM   =  -1408.929

AT ITERATION NUMBER 10 AND CALL NUMBER 443
CURRENT  FCOST =  -1408.634
CURRENT  ALM   =  -1408.958

AT ITERATION NUMBER 11 AND CALL NUMBER 480
CURRENT  FCOST =  -1408.864
CURRENT  ALM   =  -1408.933

AT ITERATION NUMBER 12 AND CALL NUMBER 517
CURRENT  FCOST =  -1409.094
CURRENT  ALM   =  -1408.939

AT ITERATION NUMBER 13 AND CALL NUMBER 554
CURRENT  FCOST =  -1408.864
CURRENT  ALM   =  -1408.946

AT ITERATION NUMBER 14 AND CALL NUMBER 573
CURRENT  FCOST =  -1408.864
CURRENT  ALM   =  -1408.884

THE FINAL FUNCTION VALUE IS (m/s): 1408.864
THE  6 VARIABLE VALUES ARE (cm & gm):
        X(1) =      3.22500
        X(2) =      0.01080
        X(3) =      0.02080
        X(4) =      0.98709
        X(5) =      0.25072
        X(6) =    891.00050
THE TOTAL NUMBER OF FUNCTION CALLS WAS :  573
THE FINAL ALM FUNCTION VALUE WAS       :  -1408.884
```

Figure 5.1.1 Example 1a ALM Iteration History.



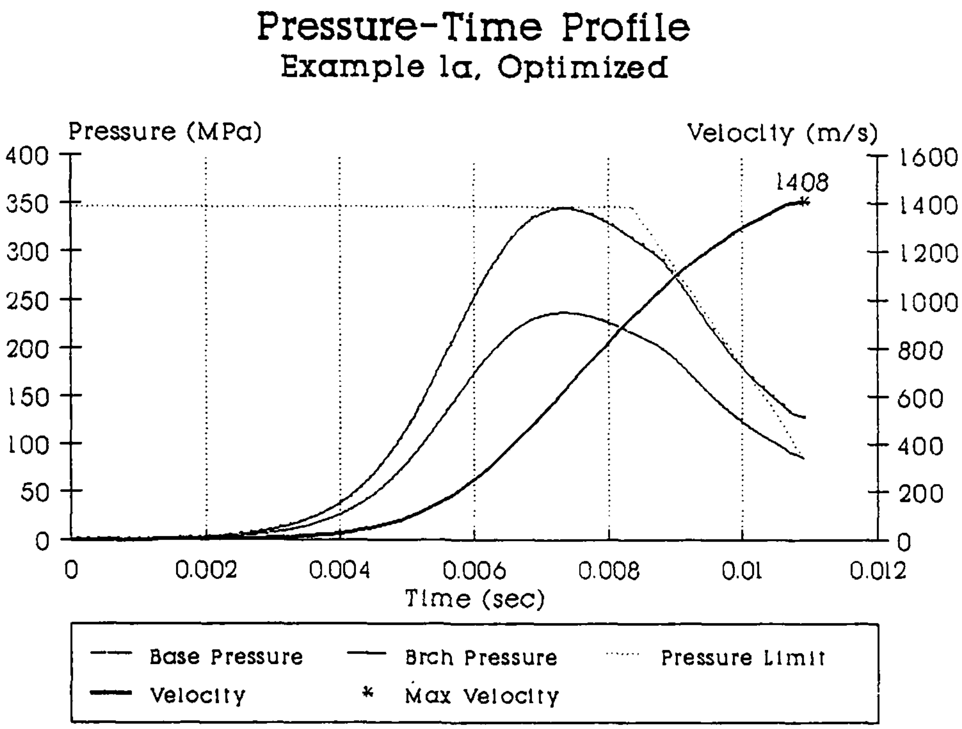
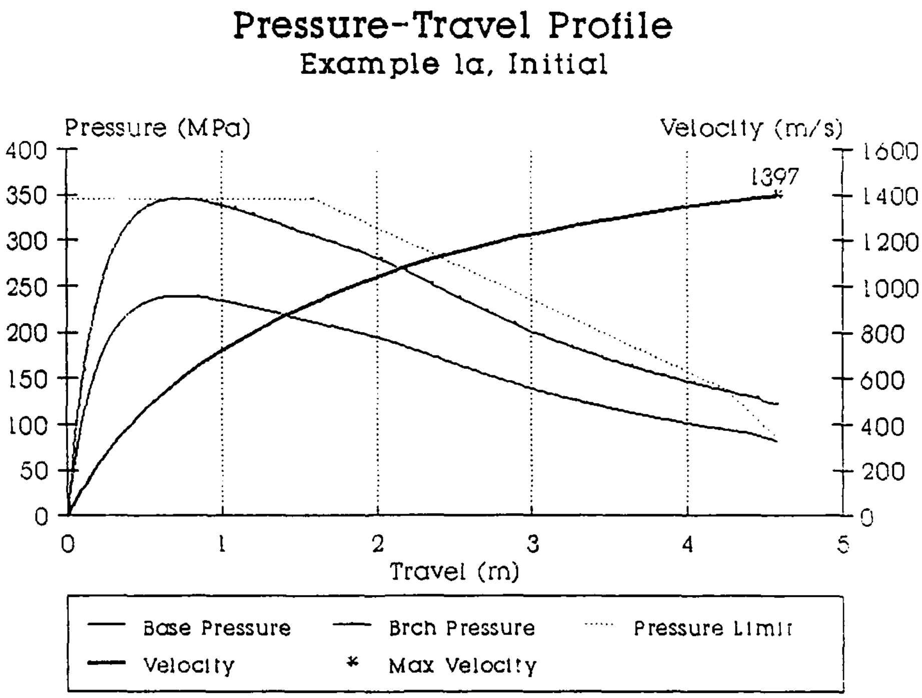


Table 5.1.2 Example 1a Pressure-Time Profiles.



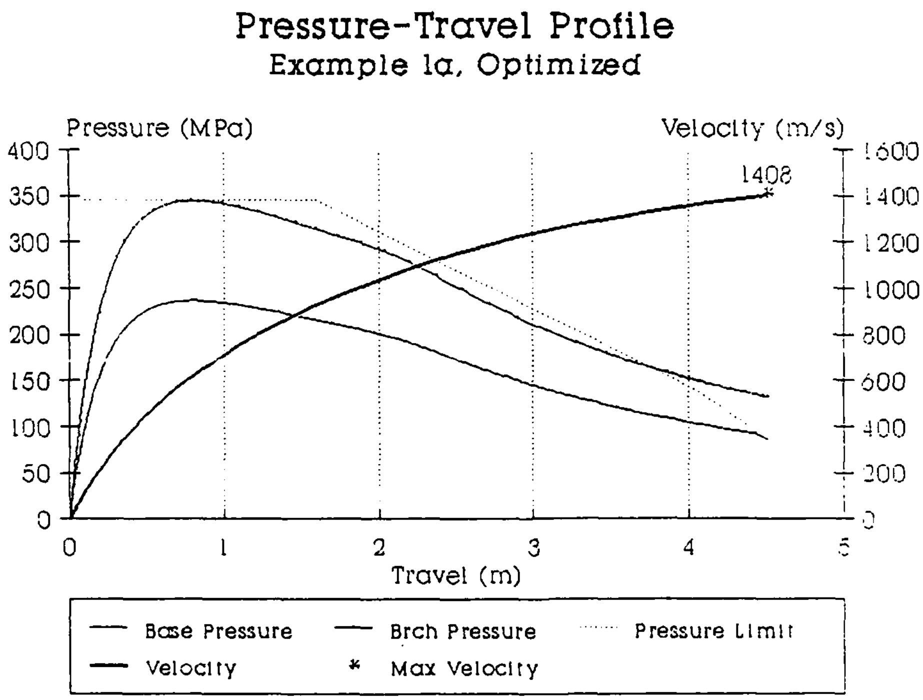


Figure 5.1.3 Example 1a Pressure-Travel Profiles.

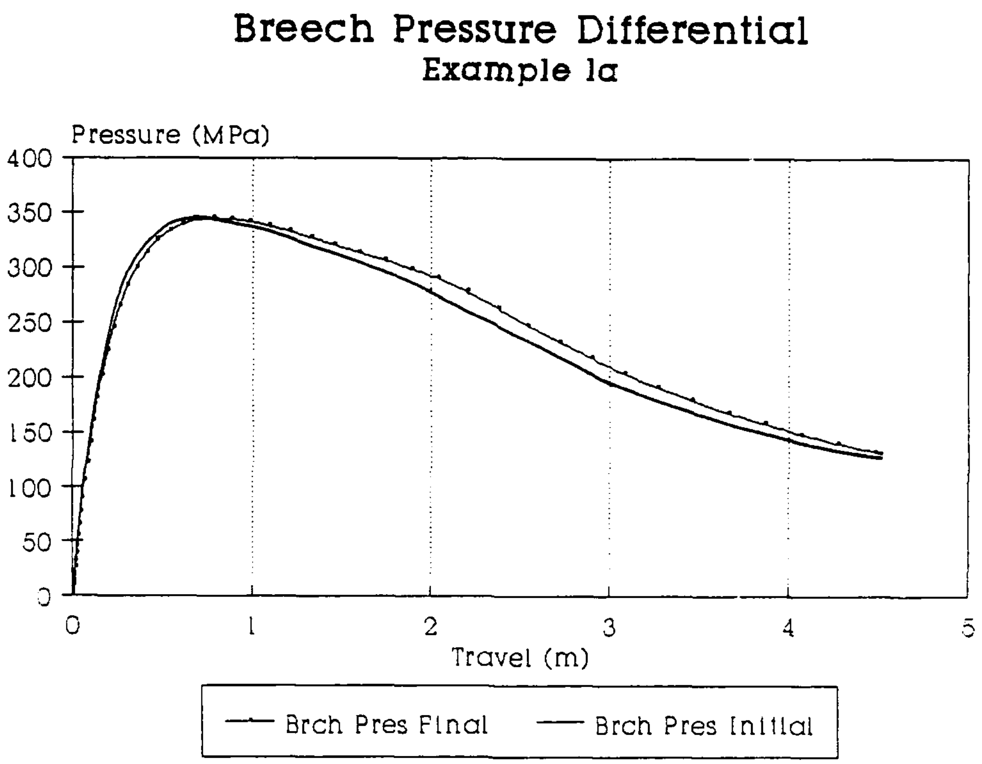


Figure 5.1.4 Example 1a Breech Pressure Differential.

Example 2: This example expands the parametric space of the first problem to include both a one and a seven perforation propellent. This gradual increase allows analysis of the optimization method in steps. A second optimization is started from the final design of the first optimization. This will check the ability of the optimization process in finding the best design. The initial and final design results are given in Table 5.2.1. The initial and final performance values of the optimization are in Table 5.2.2. The set of constraints are stated in Table 5.2.3. The ALM iteration histories for both parts of Example 2 are given in Figure 5.2.1 and 5.2.2. The pressure-time and pressure-travel profiles for Example 2 are Figure 5.2.3 and 5.2.4. The pressure differential curve is Figure 5.2.5.

The parametric space for this problem consists ten variables, the five critical dimensions for the seven perforation propellent, the three critical dimensions of the one perforation propellent, and their masses. The design vector $\overline{X}$ for Example 2 is

$$
\begin{aligned}
x_1 &= L_1, \\
x_2 &= P_{i1}, \\
x_3 &= P_{o1}, \\
x_4 &= D_1, \\
x_5 &= d_1, \\
x_6 &= L_2, \\
x_7 &= P_{i2}, \\
x_8 &= D_2, \\
x_9 &= \mathrm{mass}_1, \\
x_{10} &= \mathrm{mass}_2.
\end{aligned}
$$

The propellent dimension terms are defined at the beginning of the thesis. The subscript 1 indicates the seven perforation and the 2 indicates the one perforation propellent. There is one equality constraint and 19 inequality constraints.

Initial Values	Example 2 Propellent 1	Propellent 2
Type	Sample	Sample
No. Perf	7	1
Mass (kg)	4.35	4.35
Dimensions (cm)		
L	3.175	3.175
D	1.702	1.702
p_i	.0508	.0508
p_o	.0508	----
d	.2807	----

Final Values	Example 2 Propellent 1	Propellent 1
Type	Sample	Sample
No. Perf	7	1
Mass (kg)	4.22	3.85
Dimensions (cm)		
L	5.560	5.240
D	.8321	.4946
p_i	.0183	.0000
p_o	.0008	---
d	.2182	---

Table 5.2.1 Initial/Final Propellent Values.

Example 2

	Initial	Intermediate	Final
Projectile Velocity (m/s)	1104	1397	1430
Max Breech Pressure (MPa)	211	346	340
Max Base Pressure (MPa)	142	236	235

Table 5.2.2 Initial/Final Performance Values.

```
C***********************************************************************
      SUBROUTINE FUN_CON(X,NOVAR)
C***********************************************************************
C  THIS IS THE CONSTRAINT SET FOR EXAMPLE 2.          FH = 1     FG = 19
%INCLUDE 'declarations.ins.f'
      COMMON/limits/dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,total_vol_prop,
     +       cham_vol
      REAL*4 X(NOVAR),dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,pmax,cham_vol,
     +       total_vol_prop

C  FOR 7 PERF PROPELLENT.............................................
c  fg1  is the prop grain length .GT. 0 constraint
c  fg2  is the inner perf diam .GT. 0 constraint
c  fg3  is the outer perf diam .GT. 0 constraint
c  fg4  is the prop grain diam .GT. 0 constraint
c  fg5  is the dist between perf centers .GT. 0 constraint
c  fg6  is the prop diam .GT. (inner+outer perf diams) constraint
c  fg7  is the dist between perf centers .GT. (inner + outer radius) constraint
c  fg8  is the length .GT. diameter constraint
c  fg9  is the max length for the cord 6cm.
c  fg10 is the equilateral triangle requirement
      FG(1) =  1000*(-x(1))
      FG(2) =  1000*(-x(2))
      FG(3) =  1000*(-x(3))
      FG(4) =  1000*(-x(4))
      FG(5) =  1000*(-x(5))
      FG(6) =  100*(2*x(3)/x(4) + x(2)/x(4) - 1.0)
      FG(7) =  100*(.5*x(3)/x(5) + .5*x(2)/x(5) - 1.0)
      FG(8) =  100*(x(4) - x(1))
      FG(8) =  100*(x(4)/x(1) - 1.0)
      FG(9) =  100*(x(1)/.06 - 1.0)
      FG(10)=  100*(x(5)/x(3) - 1.0)

C  FOR 1 PERF PROPELLENTS IS ...................................
c  fg11 is the .gt. zero for length
c  fg12 is the .gt. zero for perforation
c  fg13 is the .gt. zero for diameter
c  fg14 is the max diameter constraint
c  fg15 is the perf size must be less than the diameter
c  fg16 is that the length cannot be less than the diameter
c  fg17 is the max length for the cord 6cm.
      FG(11) = 1000.*(-x(6))
      FG(12) = 1000.*(-x(7))
      FG(13) = 1000.*(-x(8))
      FG(14) = 100.*(x(8)/.04 - 1.0)
      FG(15) = 100.*(x(7)/x(8) - 1.0)
      FG(16) = 100.*(x(8)/x(6) - 1.0)
      FG(17) = 100.*(x(6)/.06 - 1.0)

c  Determine acceptable pressures.................................
c  fh1  is the max base pressure constraint
c  fg18 is the max brch pressure constraint
      pmax = 3.46e8
      FH(1)= pmaxbr/pmax - 1.0

      if (dpmaxba.gt.1.5.and.dpmaxba.le.4.0) then
        pmax = -8.320e7*dpmaxba + 4.708e8
      else if (dpmaxba.gt.4.0.and.dpmaxba.le.4.57) then
        pmax = -9.123e7*dpmaxba + 5.029e8
      else if (dpmaxba.gt.4.57) then
        pmax = .86e8
      end if
      FG(18)= 5*(pmaxba/pmax - 1.0)

C  MASS CONSTRAINT................................................
      FG(19)= 100*(total_vol_prop/cham_vol - 1.0)

      RETURN
      END
```

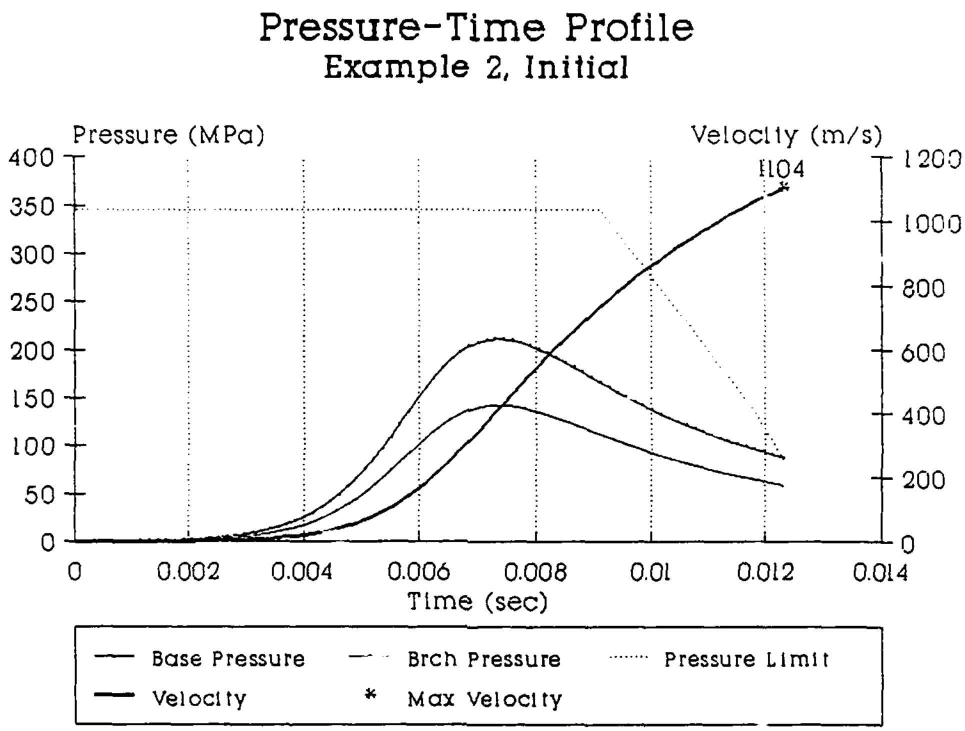
Table 5.2.3 Example 2 Constraint Set.

```
NUMBER OF DESIGN VARIABLES       : 10
NUMBER OF EQUALITY CONSTRAINTS   : 1
NUMBER OF INEQUALITY CONSTRAINTS : 19

YOU HAVE SELECTED HOOKE-JEEVES

   SEARCH DELTA = 1.0000000E-04
   ACCEL FACTOR = 2.500000

AT ITERATION NUMBER 1 AND CALL NUMBER 80
CURRENT  FCOST =  -1329.755
CURRENT  ALM   =  -1329.787

AT ITERATION NUMBER 2 AND CALL NUMBER 160
CURRENT  FCOST =  -1466.381
CURRENT  ALM   =  -1423.137

AT ITERATION NUMBER 3 AND CALL NUMBER 241
CURRENT  FCOST =  -1354.288
CURRENT  ALM   =  -1346.637

AT ITERATION NUMBER 4 AND CALL NUMBER 352
CURRENT  FCOST =  -1389.058
CURRENT  ALM   =  -1360.571

AT ITERATION NUMBER 5 AND CALL NUMBER 435
CURRENT  FCOST =  -1353.227
CURRENT  ALM   =  -1365.933

AT ITERATION NUMBER 6 AND CALL NUMBER 489
CURRENT  FCOST =  -1381.153
CURRENT  ALM   =  -1373.239

AT ITERATION NUMBER 7 AND CALL NUMBER 575
CURRENT  FCOST =  -1373.145
CURRENT  ALM   =  -1371.509

AT ITERATION NUMBER 8 AND CALL NUMBER 662
CURRENT  FCOST =  -1378.692
CURRENT  ALM   =  -1380.083

AT ITERATION NUMBER 9 AND CALL NUMBER 779
CURRENT  FCOST =  -1395.569
CURRENT  ALM   =  -1393.360

AT ITERATION NUMBER 10 AND CALL NUMBER 836
CURRENT  FCOST =  -1391.755
CURRENT  ALM   =  -1393.354

THE FINAL FUNCTION VALUE IS (m/s): 1391.755
THE  10 VARIABLE VALUES ARE (m & kg):
    X( 1) =    0.031200
    X( 2) =    0.000183
    X( 3) =    0.000458
    X( 4) =    0.009546
    X( 5) =    0.002532
    X( 6) =    0.027800
    X( 7) =    0.000058
    X( 8) =    0.006571
    X( 9) =    4.360002
    X(10) =    4.144999

THE TOTAL NUMBER OF FUNCTION CALLS WAS :  836
THE FINAL ALM FUNCTION VALUE WAS       : -1393.354
```

Figure 5.2.1 Example 2, Part 1 ALM Iteration History.

```
NUMBER OF DESIGN VARIABLES       : 10
NUMBER OF EQUALITY CONSTRAINTS   : 1
NUMBER OF INEQUALITY CONSTRAINTS : 19

YOU HAVE SELECTED POWELLS METHOD

AT ITERATION NUMBER 1 AND CALL NUMBER 81
CURRENT  FCOST =  -1564.146
CURRENT  ALM   =  -1528.375

AT ITERATION NUMBER 2 AND CALL NUMBER 134
CURRENT  FCOST =  -1548.266
CURRENT  ALM   =  -1465.383


AT ITERATION NUMBER 3 AND CALL NUMBER 215
CURRENT  FCOST =  -1447.220
CURRENT  ALM   =  -1404.940

AT ITERATION NUMBER 4 AND CALL NUMBER 272
CURRENT  FCOST =  -1418.827
CURRENT  ALM   =  -1399.732

AT ITERATION NUMBER 5 AND CALL NUMBER 331
CURRENT  FCOST =  -1403.700
CURRENT  ALM   =  -1401.875

AT ITERATION NUMBER 6 AND CALL NUMBER 447
CURRENT  FCOST =  -1421.262
CURRENT  ALM   =  -1420.808

AT ITERATION NUMBER 7 AND CALL NUMBER 616
CURRENT  FCOST =  -1427.592
CURRENT  ALM   =  -1428.370

AT ITERATION NUMBER 8 AND CALL NUMBER 790
CURRENT  FCOST =  -1430.584
CURRENT  ALM   =  -1429.993

AT ITERATION NUMBER 9 AND CALL NUMBER 935
CURRENT  FCOST =  -1430.577
CURRENT  ALM   =  -1430.759

AT ITERATION NUMBER 10 AND CALL NUMBER 1024
CURRENT  FCOST =  -1430.334
CURRENT  ALM   =  -1430.790

THE FINAL FUNCTION VALUE IS (m/s): 1430.334
THE  10 VARIABLE VALUES ARE (m & kg):
    X( 1) =   0.055600
    X( 2) =   0.000183
    X( 3) =   0.000008
    X( 4) =   0.008321
    X( 5) =   0.002182
    X( 6) =   0.052400
    X( 7) =   0.000000
    X( 8) =   0.004946
    X( 9) =   4.217497
    X(10) =   3.849896

THE TOTAL NUMBER OF FUNCTION CALLS WAS :  1024
THE FINAL ALM FUNCTION VALUE WAS       : -1430.790
```

Figure 5.2.2 Example 2, Part 2 ALM Iteration History.



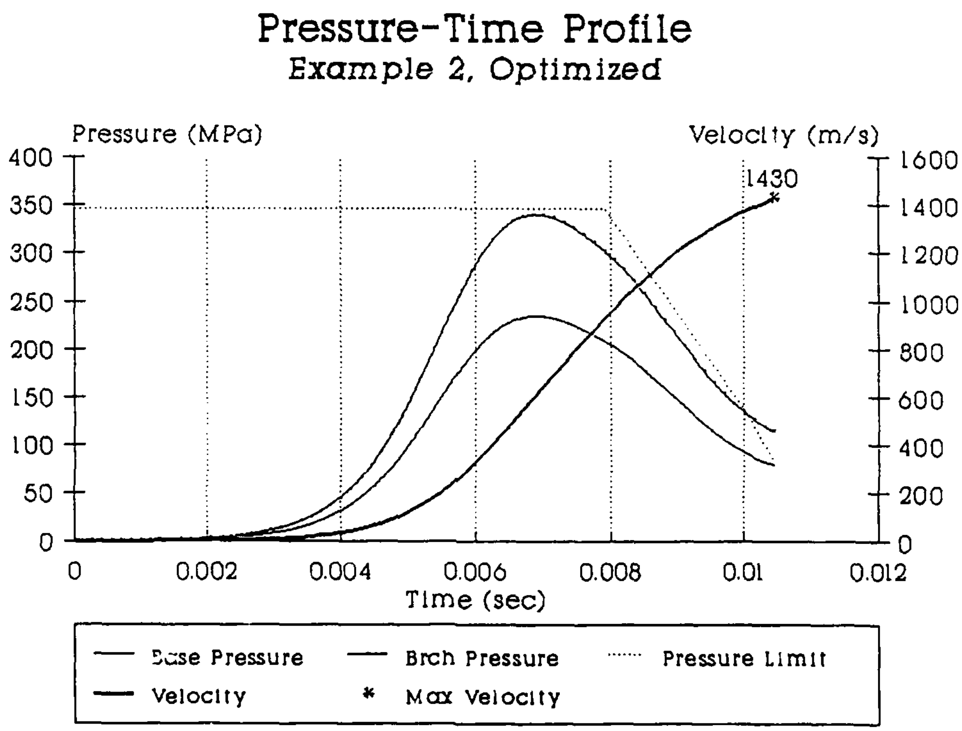
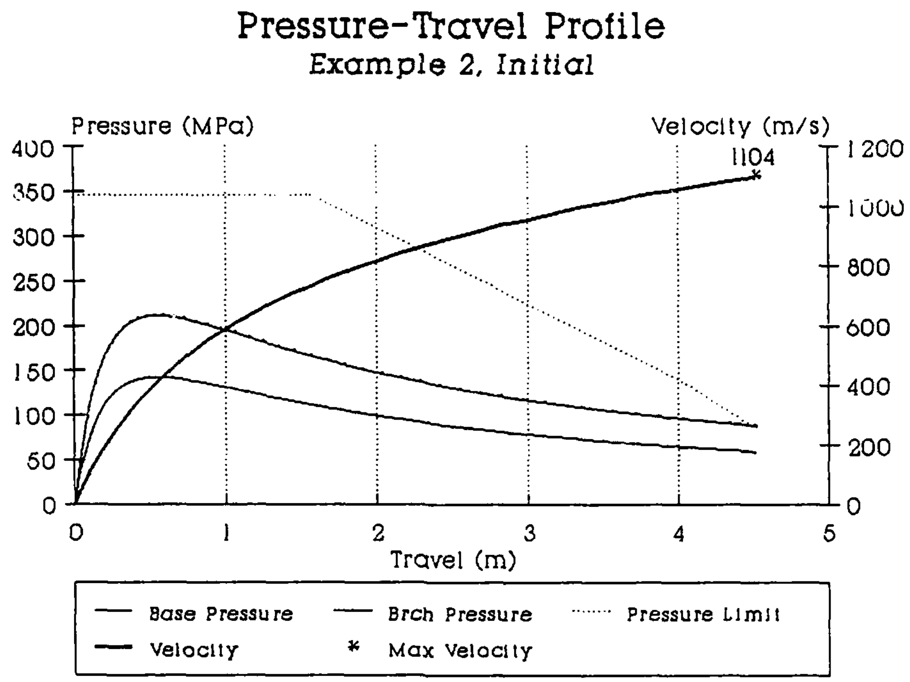


Figure 5.2.3 Example 2 Pressure-Time Profiles.



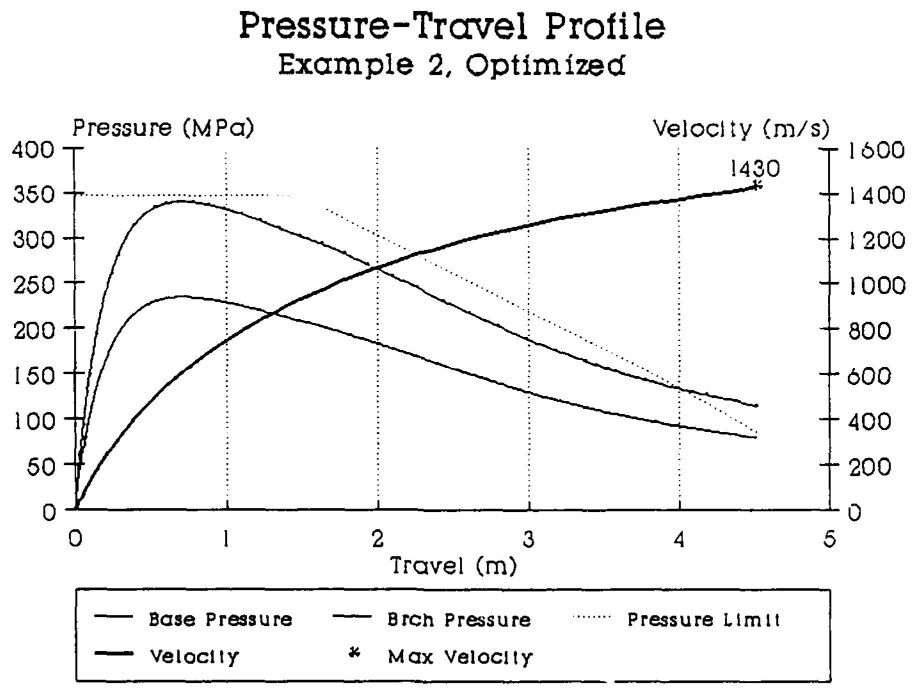


Figure 5.2.4 Example 2 Pressure-Travel Profiles.

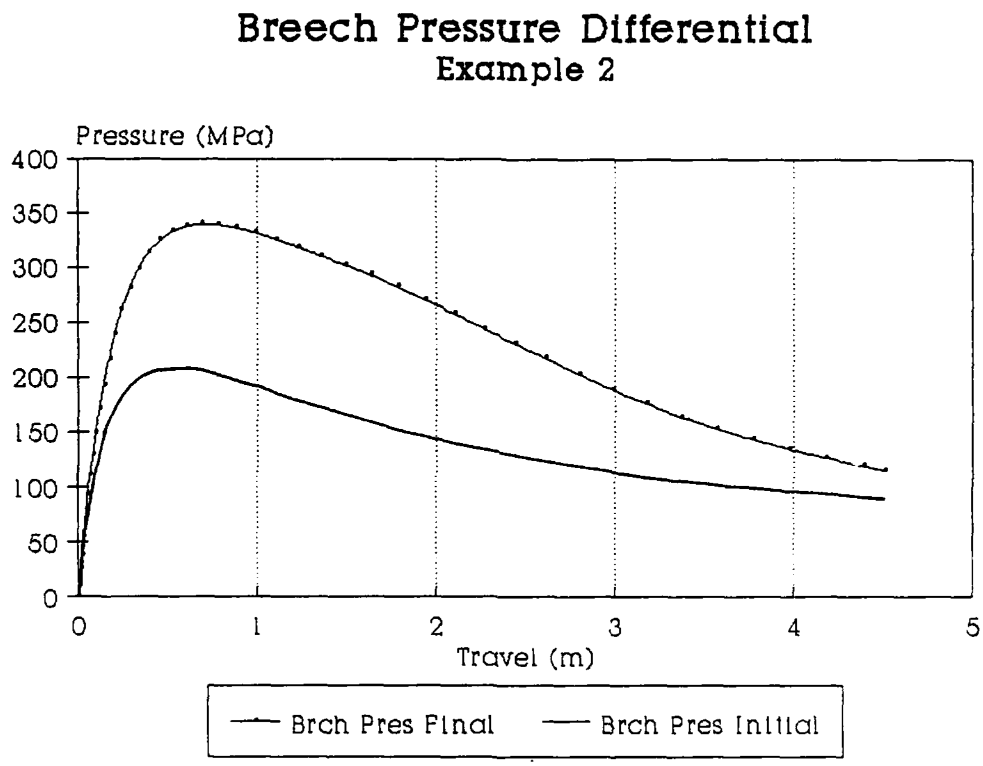


Figure 5.2.5 Example 2 Breech Pressure Differential.

Example 3: This example further expands the parametric space of the first and second problem to include a zero, one, and seven perforation propellent. This final increase further examines the performance of the optimization method in an even larger parametric space. The initial and final design results are given in Table 5.3.1. The initial and final performance values of the optimization are in Table 5.3.2. The set of constraints are stated in Table 5.3.3. The ALM iteration history for Example 3 is given in Figure 5.3.1. The pressure-time and pressure-travel profiles for Example 3 are Figure 5.3.2 and 5.3.3. The pressure differential curve is Figure 5.3.4.

The parametric space for this problem consists thirteen variables, the two critical dimensions for the zero perforation propellent, the three critical dimensions of the one perforation propellent, the five critical dimensions for the seven perforation propellent, and their masses. The design vector $\overline{X}$ for Example 3 is

$$
\begin{aligned}
x_1 &= L_1, \\
x_2 &= D_1, \\
x_3 &= L_2, \\
x_4 &= P_{i2}, \\
x_5 &= D_2, \\
x_6 &= L_3, \\
x_7 &= P_{i3}, \\
x_8 &= P_{o3}, \\
x_9 &= D_3, \\
X_{10} &= d_3, \\
x_{11} &= \mathrm{mass}_1, \\
x_{12} &= \mathrm{mass}_2, \\
x_{13} &= \mathrm{mass}_3.
\end{aligned}
$$

The propellent dimension terms are defined at the beginning of the thesis. The subscript 1 indicates the zero perforation, the 2 indicates the one perforation propellent, and the 3 indicates the seven perforation propellent. There is one equality constraint and 25 inequality constraints.

Initial Values	Propellent 1	Example 3 Propellent 2	Propellent 3
Type	Sample	Sample	Sample
No. Perf	0	1	7
Mass (kg)	3.00	3.00	3.00
Dimensions (cm)			
L	3.175	3.175	3.175
D	1.702	1.702	1.702
p_i	---	.0508	.0508
p_o	---	---	.0508
d	---	---	.2807

Final Values	Propellent 1	Example 3 Propellent 2	Propellent 3
Type	Sample	Sample	Sample
No. Perf	0	1	7
Mass (kg)	.99	3.63	4.13
Dimensions (cm)			
L	2.494	5.943	5.996
D	1.285	.580	5.996
p_i	---	.0000	.0018
p_o	---	---	.0018
d	---	---	.2170

Table 5.3.1 Initial/Final Propellent Values.

Example 3

	Initial	Final
Projectile Velocity (m/s)	1049	1368
Max Breech Pressure (MPa)	218	346
Max Base Pressure (MPa)	149	239

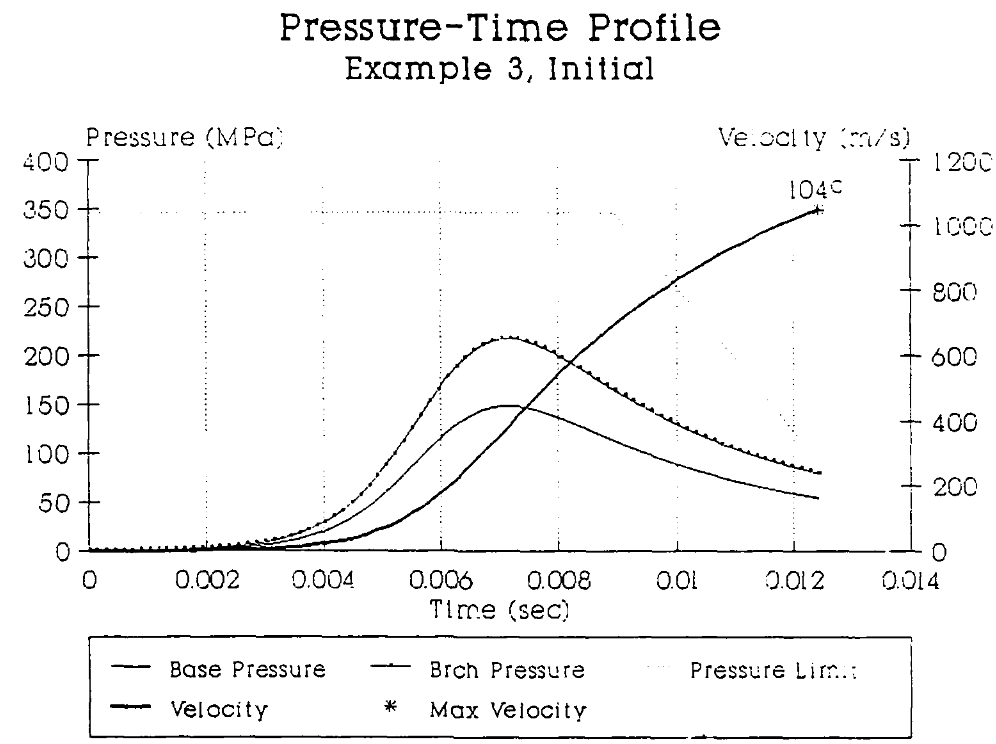
Table 5.3.2 Initial/Final Performance Values.

```
C***********************************************************************
      SUBROUTINE FUN_CON(X,NOVAR)
C***********************************************************************
C  THIS IS THE CONSTRAINT SET FOR EXAMPLE 3.        FH =  1      FG = 25
%INCLUDE 'declarations.ins.f'
      COMMON/limits/dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,total_vol_prop,
     +       cham_vol
      REAL*4 X(NOVAR),dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,pmax,cham_vol,
     +       total_vol_prop

C  FOR 0 PERF PROPELLENT..................................dimensions.........
c  fg1  :prop grain length .GT. 0 constraint                  m
c  fg2  :prop grain diam .GT. 0 constraint                    m
c  fg3  :mass .GT. than 0 constraint                          kg
c  fg4  :length .GT. diameter constraint
c  fg5  :max length for the cord 6cm.
      FG(1) = 1000*(-x(1))
      FG(2) = 1000*(-x(2))
      FG(3) =  100*(-x(11))
      FG(4) =  100*(x(2)/x(1) - 1.0)
      FG(5) =  100*(x(1)/.06 - 1.0)

C  FOR 1 PERF PROPELLENT..................................dimensions.........
c  fg6  :prop grain length .GT. 0 constraint                  m
c  fg7  :inner perf diam .GT. 0 constraint                    m
c  fg8  :prop grain diam .GT. 0 constraint                    m
c  fg9  :mass .GT. than 0 constraint                          kg
c  fg10 :prop diam .GT. inner perf diam constraint
c  fg11 :length .GT. diameter constraint
c  fg12 :max length for the cord 6cm.
      FG(6)  = 1000*(-x(3))
      FG(7)  = 1000*(-x(4))
      FG(8)  = 1000*(-x(5))
      FG(9)  =  100*(-x(12))
      FG(10) =  100*(x(4)/x(5) - 1.0)
      FG(11) =  100*(x(5)/x(3) - 1.0)
      FG(12) =  100*(x(3)/.06 - 1.0)

C  FOR 7 PERF PROPELLENT..................................dimensions.........
c  fg13  :prop grain length .GT. 0 constraint                 m
c  fg14  :inner perf diam .GT. 0 constraint                   m
c  fg15  :outer perf diam .GT. 0 constraint                   m
c  fg16  :prop grain diam .GT. 0 constraint                   m
c  fg17  :dist between perf centers .GT. 0 constraint         m
c  fg18  :mass .GT. than 0 constraint                         kg
c  fg19  :prop diam .GT. (inner+outer perf diams) constraint
c  fg20  :dist between perf centers .GT. (inner + outer radius) constraint
c  fg21  :length .GT. diameter constraint
c  fg22  :max length for the cord 6cm.
c  fg23  :equilateral triangle requirement.
c  fg24  :max base pressure constraint
c  fg25  :maximum volume of propellent cannot exceed the space in the chamber
      FG(13) = 1000*(-x(6))
      FG(14) = 1000*(-x(7))
      FG(15) = 1000*(-x(8))
      FG(16) = 1000*(-x(9))
      FG(17) = 1000*(-x(10))
      FG(18) =  100*(-x(13))
      FG(19) =  100*(2*x(8)/x(9) + x(7)/x(9) - 1.0)
      FG(20) =  100*(.5*x(8)/x(10) + .5*x(7)/x(10) - 1.0)
      FG(21) =  100*(x(9)/x(6) - 1.0)
      FG(22) =  100*(x(6)/.06 - 1.0)
      FG(23) =  100*(3.0*x(8)/(4.0*x(9)) + x(7)/(4.0*x(9)) - 1.0)

c  fh1  :max breech pressure constraint
      pmax  = 3.46e8
      FH(1) =  pmaxbr/pmax - 1.0
      if (dpmaxba.gt.1.5.and.dpmaxba.le.4.0) then
        pmax = -8.320e7*dpmaxba + 4.708e8
      else if (dpmaxba.gt.4.0.and.dpmaxba.le.4.57) then
        pmax = -9.123e7*dpmaxba + 5.029e8
      else if (dpmaxba.gt.4.57) then
        pmax = .86e8
      end if
      FG(24) =   5*(pmaxba/pmax - 1.0)
      FG(25) = 100*(total_vol_prop/cham_vol - 1.0)

      RETURN
      END
```

Table 5.3.3 Example 3 Constraint Set.

```
NUMBER OF DESIGN VARIABLES     : 13
NUMBER OF EQUALITY CONSTRAINTS: 1
NUMBER OF EQUALITY CONSTRAINTS: 25

YOU HAVE SELECTED HOOKE-JEEVES

   SEARCH DELTA = 1.0000000E-04
   ACCEL FACTOR = 2.500000

AT ITERATION NUMBER 1 AND CALL NUMBER 101
CURRENT  FCOST =  -1491.332
CURRENT  ALM   =  -1461.602

AT ITERATION NUMBER 2 AND CALL NUMBER 167
CURRENT  FCOST =  -1467.293
CURRENT  ALM   =  -1404.639

AT ITERATION NUMBER 3 AND CALL NUMBER 236
CURRENT  FCOST =  -1417.090
CURRENT  ALM   =  -1361.723

AT ITERATION NUMBER 4 AND CALL NUMBER 306
CURRENT  FCOST =  -1366.709
CURRENT  ALM   =  -1355.552

AT ITERATION NUMBER 5 AND CALL NUMBER 375
CURRENT  FCOST =  -1361.439
CURRENT  ALM   =  -1357.584

AT ITERATION NUMBER 6 AND CALL NUMBER 519
CURRENT  FCOST =  -1364.183
CURRENT  ALM   =  -1365.158

AT ITERATION NUMBER 7 AND CALL NUMBER 664
CURRENT  FCOST =  -1365.460
CURRENT  ALM   =  -1366.705

AT ITERATION NUMBER 8 AND CALL NUMBER 777
CURRENT  FCOST =  -1367.985
CURRENT  ALM   =  -1367.011

AT ITERATION NUMBER 9 AND CALL NUMBER 853
CURRENT  FCOST =  -1367.394
CURRENT  ALM   =  -1366.895

AT ITERATION NUMBER 10 AND CALL NUMBER 930
CURRENT  FCOST =  -1366.816
CURRENT  ALM   =  -1366.905

AT ITERATION NUMBER 11 AND CALL NUMBER 1121
CURRENT  FCOST =  -1367.399
CURRENT  ALM   =  -1367.278

AT ITERATION NUMBER 12 AND CALL NUMBER 1316
CURRENT  FCOST =  -1367.478
CURRENT  ALM   =  -1367.605

AT ITERATION NUMBER 13 AND CALL NUMBER 1394
CURRENT  FCOST =  -1367.565
CURRENT  ALM   =  -1367.590

THE FINAL FUNCTION VALUE IS (m/s): 1367.565
THE  13 VARIABLE VALUES ARE (m & kg):
    X( 1) =   0.024935
    X( 2) =   0.012850
    X( 3) =   0.059425
    X( 4) =   0.000000
    X( 5) =   0.005798
    X( 6) =   0.059960
    X( 7) =   0.000018
    X( 8) =   0.000018
    X( 9) =   0.008613
    X(10) =   0.002171
    X(11) =   0.992000
    X(12) =   3.625999
    X(13) =   4.130000

THE TOTAL NUMBER OF FUNCTION CALLS WAS :  1394
THE FINAL ALM FUNCTION VALUE WAS       :  -1367.590
```

Figure 5.3.1 Example 3 ALM Iteration History.



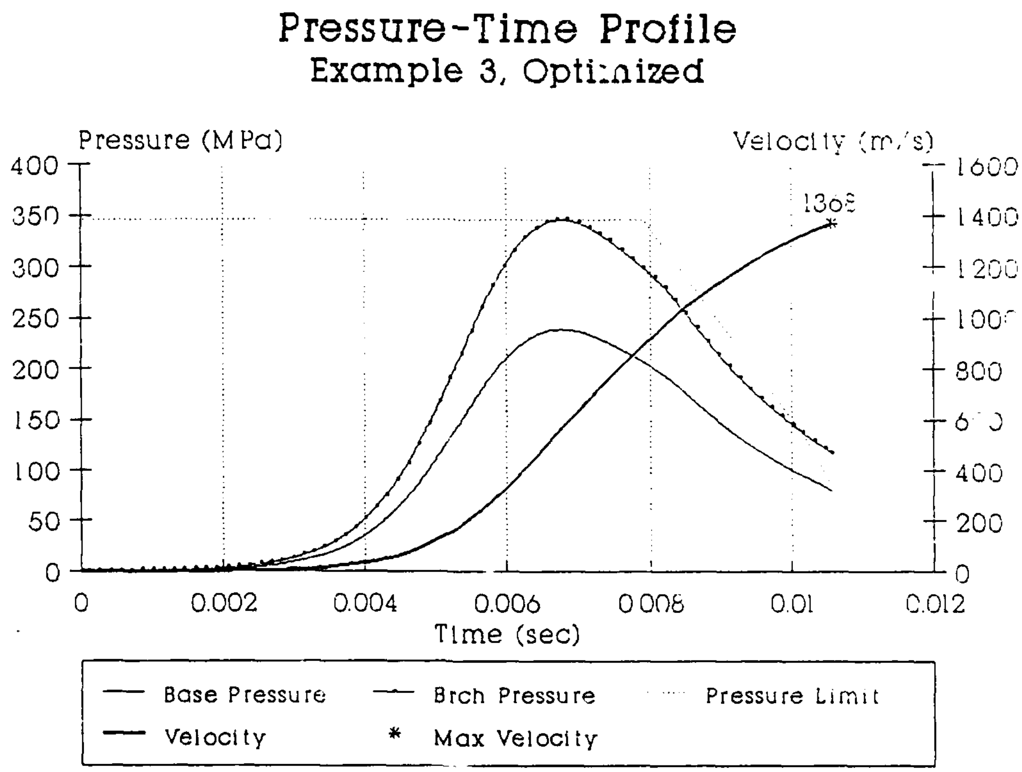
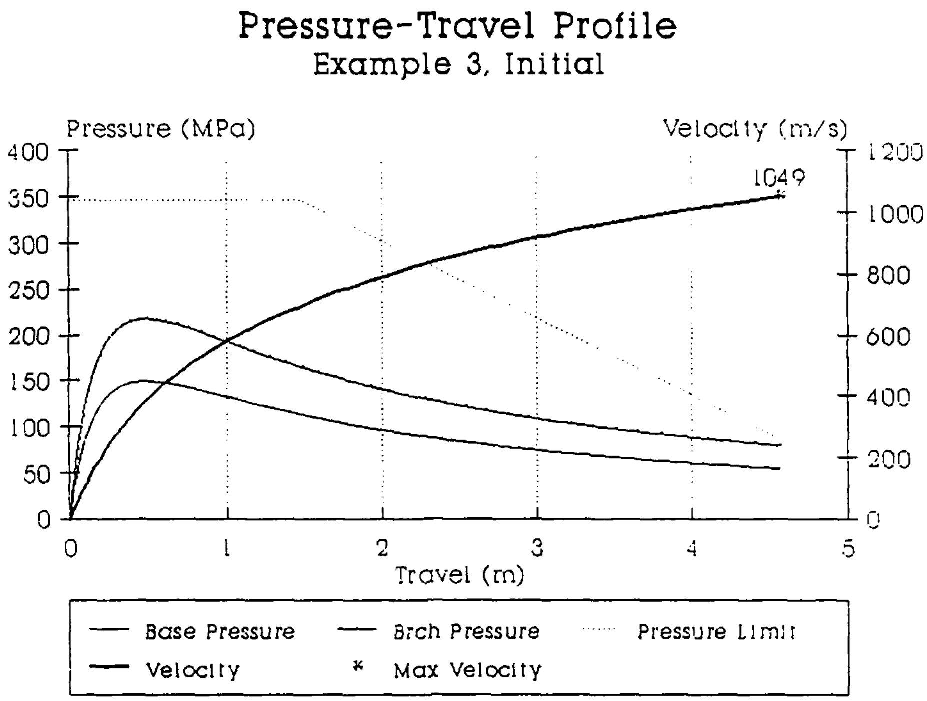


Figure 5.3.2 Example 3 Pressure-Time Profiles.



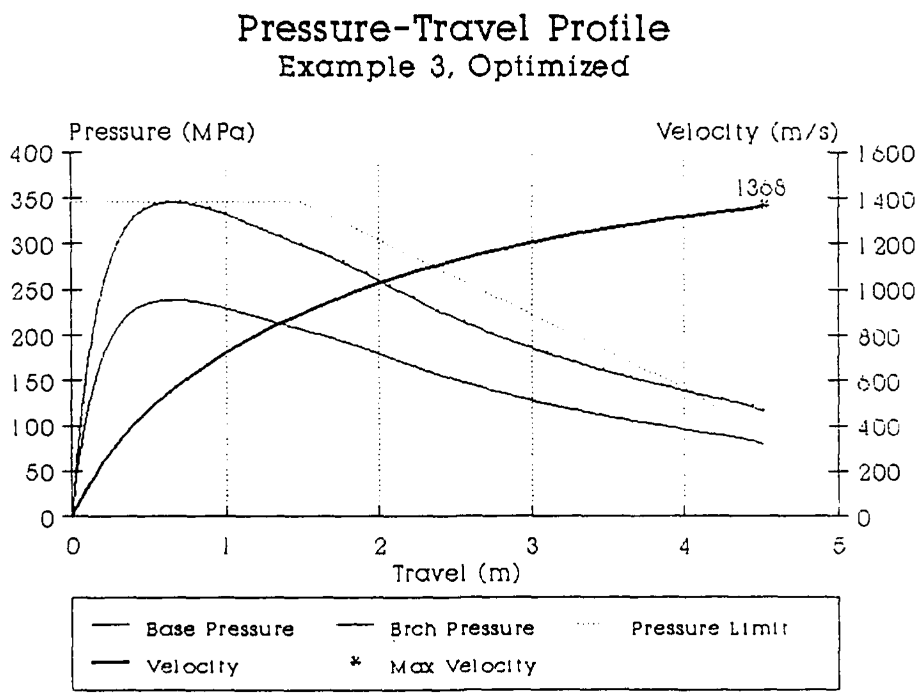


Figure 5.3.3 Example 3 Pressure-Travel Profiles.

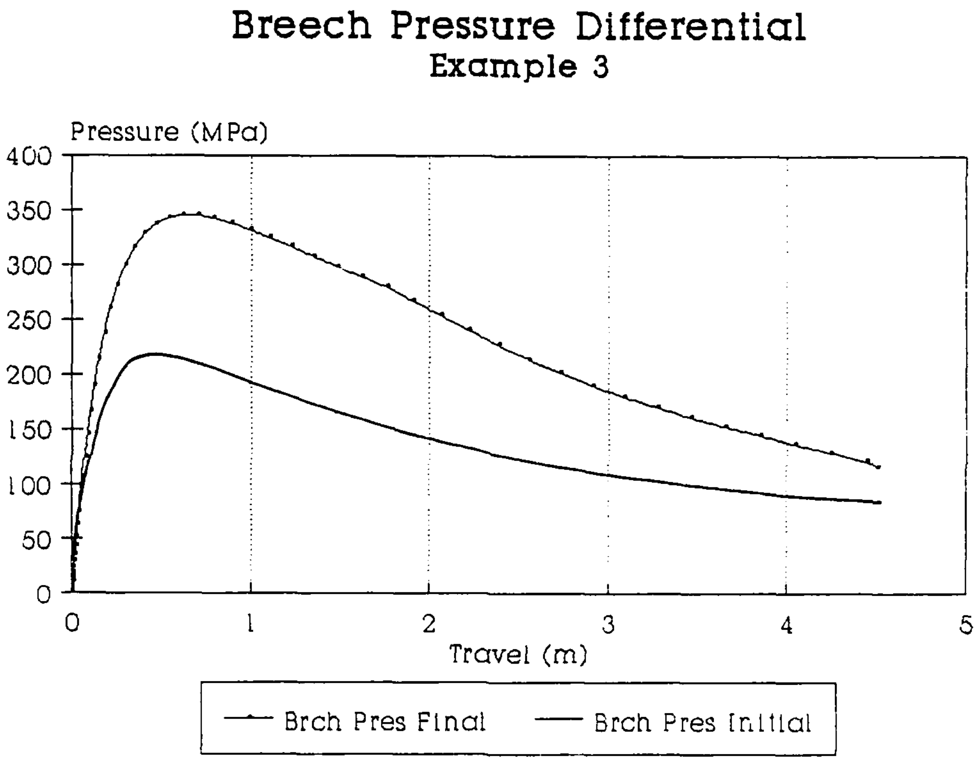


Figure 5.3.4 Example 3 Breech Pressure Differential.

Example 4: This example demonstrates the optimization of two propellents with different thermodynamic characteristics. Two seven perforations propellents are used. This allows the comparison of the optimization method in a different parametric space. Just like the second example, a second optimization is started from the final design of the first optimization. This checks the ability of the optimization process in finding the best design. The initial and final design results are given in Table 5.4.1. The initial and final performance values of the optimization are in Table 5.4.2. The set of constraints are stated in Table 5.4.3. The ALM iteration histories for both parts of Example 4 are given in Figure 5.4.1 and 5.4.2. The pressure-time and pressure-travel profiles for Example 4 are Figure 5.4.3 and 5.4.4. The pressure differential curve is Figure 5.4.5.

The parametric space for this problem consists twelve variables, the five critical dimensions for the first seven perforation propellent, the five critical dimensions for the second seven perforation propellent, and their masses. The design vector $\bar{X}$ for Example 4 is

$$
\begin{aligned}
x_1 &= L_1, \\
x_2 &= P_{i1}, \\
x_3 &= P_{o1}, \\
x_4 &= D_1, \\
x_5 &= d_1, \\
x_6 &= L_2, \\
x_7 &= P_{i2}, \\
x_8 &= P_{o2}, \\
x_9 &= D_2, \\
x_{10} &= d_2,
\end{aligned}
$$

x_{11} = $mass_1$,
x_{12} = $mass_2$.

The propellent dimension terms are defined at the beginning of the thesis. The subscript 1 indicates the first seven perforation propellent and the 2 indicates the second seven perforation propellent. There is one equality constraint and 24 inequality constraints.

Initial Values	Example 4 Propellent 1	Propellent 2
Type	Sample	M8
No. Perf	7	7
Mass (kg)	4.35	4.35
Dimensions (cm)		
L	3.175	3.175
D	1.702	1.702
P_i	.0508	.0508
P_o	.0508	.0508
d	.2807	.2807

Final Values	Example 4 Propellent 1	Propellent 2
Type	Sample	M8
No. Perf	7	7
Mass (kg)	5.37	3.20
Dimensions (cm)		
L	3.576	3.548
D	.9821	1.002
P_i	.0800	.0383
P_o	.0400	.0508
d	.2607	.2707

Table 5.4.1 Initial/Final Propellent Values.

Example 4

	Initial	Intermediate	Final
Projectile Velocity (m/s)	1341	1391	1395
Max Breech Pressure (MPa)	325	346	345
Max Base Pressure (MPa)	226	234	234

Table 5.4.2 Initial/Final Performance Values.

```
C***********************************************************************
      SUBROUTINE FUN_CON(X,NOVAR)
C***********************************************************************
C  THIS IS THE CONSTRAINT SET FOR EXAMPLE 4          FH = 1     FG = 24
%INCLUDE 'declarations.ins.f'
      COMMON/limits/dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,total_vol_prop,
     +       cham_vol
      REAL*4 X(NOVAR),dpmaxba,dpmaxbr,pmaxbr,pmaxba,d_l,pmax,cham_vol,
     +       total_vol_prop

C  FOR SAMPLE 7 PERF PROPELLENT.......................................
c  fg1  is the prop grain length .GT. 0 constraint
c  fg2  is the inner perf diam .GT. 0 constraint
c  fg3  is the outer perf diam .GT. 0 constraint
c  fg4  is the prop grain diam .GT. 0 constraint
c  fg5  is the dist between perf centers .GT. 0 constraint
c  fg6  is the mass .GT. 0 constraint
c  fg7  is the prop diam .GT. (inner+outer perf diams) constraint
c  fg8  is the dist between perf centers .GT. (inner + outer radius) constraint
c  fg9  is the length .GT. diameter constraint
c  fg10 is the max length for the cord 6cm
c  fg11 is the equilateral triangle requirement
      FG(1) = 1000*(-x(1))
      FG(2) = 1000*(-x(2))
      FG(3) = 1000*(-x(3))
      FG(4) = 1000*(-x(4))
      FG(5) = 1000*(-x(5))
      FG(6) =  100*(-x(11))
      FG(7) =  100*(2*x(3)/x(4) + x(2)/x(4) - 1.0)
      FG(8) =  100*(.5*x(3)/x(5) + .5*x(2)/x(5) - 1.0)
      FG(9) =  100*(x(4)/x(1) - 1.0)
      FG(10)=  100*(x(1)/.06  - 1.0)
      FG(11)=  100*(x(5)/x(3) - 1.0)

C  FOR M8 7 PERF PROPELLENTS IS ......................................
c  fg12 is the prop grain length .GT. 0 constraint
c  fg13 is the inner perf diam .GT. 0 constraint
c  fg14 is the outer perf diam .GT. 0 constraint
c  fg15 is the prop grain diam .GT. 0 constraint
c  fg16 is the dist between perf centers .GT. 0 constraint
c  fg17 is the mass .GT. 0 constraint
c  fg18 is the prop diam .GT. (inner+outer perf diams) constraint
c  fg19 is the dist between perf centers .GT. (inner + outer radius) constraint
c  fg20 is the length .GT. diameter constraint
c  fg21 is the max length for the cord 6cm.
c  fg22 is the equilateral triangle requirement
c  fg23 is the max base pressure constraint
c  fh1  is the max brch pressure constraint
      FG(12)= 1000*(-x(6))
      FG(13)= 1000*(-x(7))
      FG(14)= 1000*(-x(8))
      FG(15)= 1000*(-x(9))
      FG(16)= 1000*(-x(10))
      FG(17)=  100*(-x(12))
      FG(18)=  100*(2*x(8)/x(9) + x(7)/x(9) - 1.0)
      FG(19)=  100*(.5*x(8)/x(10) + .5*x(7)/x(10) - 1.0)
      FG(20)=  100*(x(9)/x(6)- 1.0)
      FG(21)=  100*(x(6)/.06 - 1.0)
      FG(22)=  100*(x(10)/x(8) - 1.0)

c  Determine acceptable pressures.................................
      pmax = 3.46e8
      FH(1)= pmaxbr/pmax - 1.0

      if (dpmaxba.gt.1.5.and.dpmaxba.le.4.0) then
        pmax = -8.320e7*dpmaxba + 4.708e8
      else if (dpmaxba.gt.4.0.and.dpmaxba.le.4.57) then
        pmax = -9.123e7*dpmaxba + 5.029e8
      else if (dpmaxba.gt.4.57) then
        pmax = .86e8
      end if

      FG(23)= 5*(pmaxba/pmax - 1.0)
      FG(24)= 100*(total_vol_prop/cham_vol - 1.0)

      RETURN
      END
```

Table 5.4.3 Example 4 Constraint Set.

```
NUMBER OF DESIGN VARIABLES       : 12
NUMBER OF EQUALITY CONSTRAINTS   : 1
NUMBER OF INEQUALITY CONSTRAINTS : 24

YOU HAVE SELECTED HOOKE-JEEVES

   SEARCH DELTA = 1.0000000E-04
   ACCEL FACTOR = 2.500000

AT ITERATION NUMBER 1 AND CALL NUMBER 92
CURRENT  FCOST =  -1519.152
CURRENT  ALM   =  -1487.421

AT ITERATION NUMBER 2 AND CALL NUMBER 155
CURRENT  FCOST =  -1480.552
CURRENT  ALM   =  -1419.600

AT ITERATION NUMBER 3 AND CALL NUMBER 217
CURRENT  FCOST =  -1423.870
CURRENT  ALM   =  -1374.297

AT ITERATION NUMBER 4 AND CALL NUMBER 279
CURRENT  FCOST =  -1371.400
CURRENT  ALM   =  -1359.452

AT ITERATION NUMBER 5 AND CALL NUMBER 381
CURRENT  FCOST =  -1367.596
CURRENT  ALM   =  -1365.239

AT ITERATION NUMBER 6 AND CALL NUMBER 483
CURRENT  FCOST =  -1365.586
CURRENT  ALM   =  -1367.228

AT ITERATION NUMBER 7 AND CALL NUMBER 620
CURRENT  FCOST =  -1368.504
CURRENT  ALM   =  -1368.291

AT ITERATION NUMBER 8 AND CALL NUMBER 827
CURRENT  FCOST =  -1377.971
CURRENT  ALM   =  -1378.822

AT ITERATION NUMBER 9 AND CALL NUMBER 1039
CURRENT  FCOST =  -1392.342
CURRENT  ALM   =  -1391.591

AT ITERATION NUMBER 10 AND CALL NUMBER 1144
CURRENT  FCOST =  -1391.594
CURRENT  ALM   =  -1391.761

THE FINAL FUNCTION VALUE IS (m/s): 1391.594
THE  12 VARIABLE VALUES ARE (m & kg):
    X( 1) =   0.035075
    X( 2) =   0.000233
    X( 3) =   0.000458
    X( 4) =   0.009871
    X( 5) =   0.002607
    X( 6) =   0.034775
    X( 7) =   0.000983
    X( 8) =   0.000458
    X( 9) =   0.010621
    X(10) =   0.002857
    X(11) =   5.323147
    X(12) =   3.254370

THE TOTAL NUMBER OF FUNCTION CALLS WAS :  1144
THE FINAL ALM FUNCTION VALUE WAS       : -1391.761
```

Figure 5.4.1 Example 4, Part 1 ALM Iteration History.

```
NUMBER OF DESIGN VARIABLES       : 6
NUMBER OF EQUALITY CONSTRAINTS   : 1
NUMBER OF INEQUALITY CONSTRAINTS : 13

YOU HAVE SELECTED POWELLS METHOD

AT ITERATION NUMBER 1 AND CALL NUMBER 95
CURRENT  FCOST =  -1542.902
PREVIOUS FCOST =  1.0000000E-06
CURRENT  ALM   =  -1501.385
PREVIOUS ALM   =  1.0000000E-06

AT ITERATION NUMBER 2 AND CALL NUMBER 158
CURRENT  FCOST =  -1502.262
PREVIOUS FCOST =  -1542.902
CURRENT  ALM   =  -1442.825
PREVIOUS ALM   =  -1501.385

AT ITERATION NUMBER 3 AND CALL NUMBER 222
CURRENT  FCOST =  -1443.458
PREVIOUS FCOST =  -1502.262
CURRENT  ALM   =  -1402.508
PREVIOUS ALM   =  -1442.825

AT ITERATION NUMBER 4 AND CALL NUMBER 291
CURRENT  FCOST =  -1407.329
PREVIOUS FCOST =  -1443.458
CURRENT  ALM   =  -1394.635
PREVIOUS ALM   =  -1402.508

AT ITERATION NUMBER 5 AND CALL NUMBER 362
CURRENT  FCOST =  -1394.807
PREVIOUS FCOST =  -1407.329
CURRENT  ALM   =  -1393.702
PREVIOUS ALM   =  -1394.635

AT ITERATION NUMBER 6 AND CALL NUMBER 467
CURRENT  FCOST =  -1395.328
PREVIOUS FCOST =  -1394.807
CURRENT  ALM   =  -1395.892
PREVIOUS ALM   =  -1393.702

AT ITERATION NUMBER 7 AND CALL NUMBER 535
CURRENT  FCOST =  -1395.332
PREVIOUS FCOST =  -1395.328
CURRENT  ALM   =  -1396.150
PREVIOUS ALM   =  -1395.892

THE FINAL FUNCTION VALUE IS (m/s): 1395.332
THE  12 VARIABLE VALUES ARE (m & kg):
    X( 1) =   0.035760
    X( 2) =   0.000083
    X( 3) =   0.000408
    X( 4) =   0.009821
    X( 5) =   0.002607
    X( 6) =   0.035480
    X( 7) =   0.000383
    X( 8) =   0.000508
    X( 9) =   0.010021
    X(10) =   0.002707
    X(11) =   5.368001
    X(12) =   3.204000

THE TOTAL NUMBER OF FUNCTION CALLS WAS :  535
THE FINAL ALM FUNCTION VALUE WAS       : -1396.150
```

Figure 5.4.2 Example 4, Part 2 ALM Iteration History.

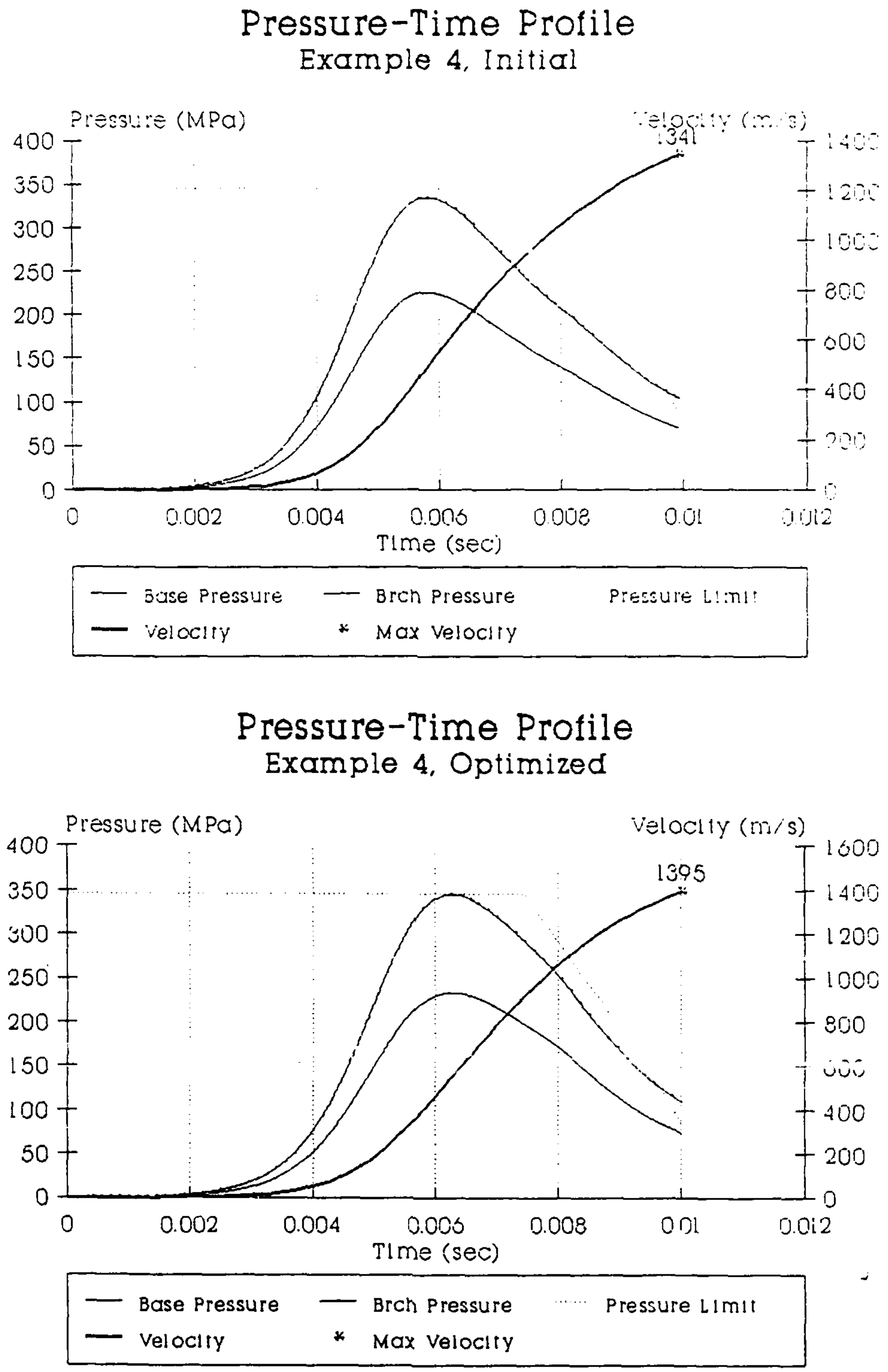
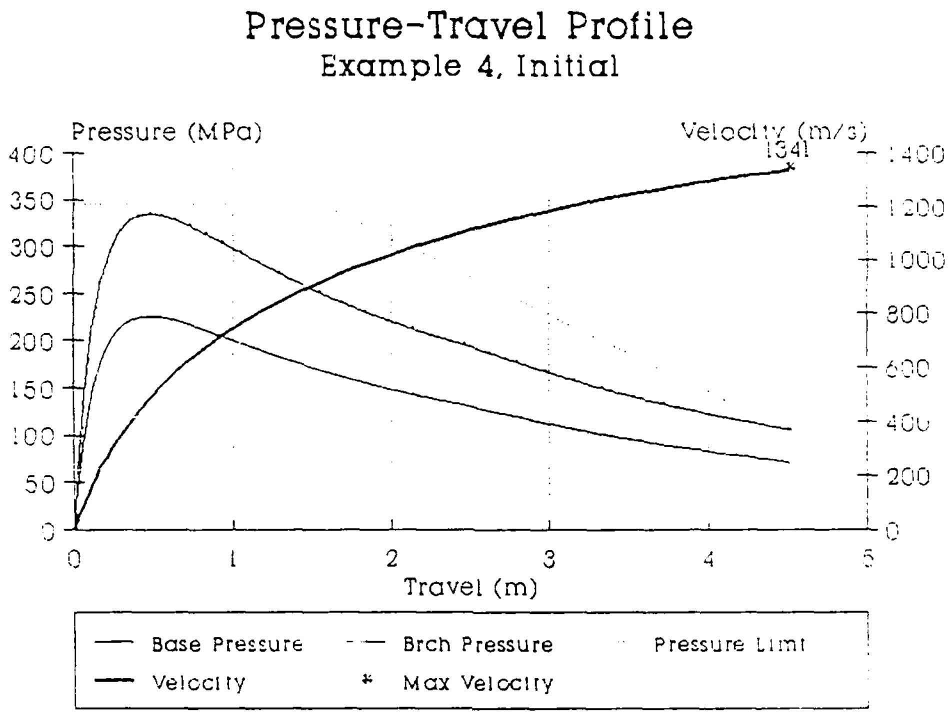


Figure 5.4.3 Example 4 Pressure-Time Profiles.



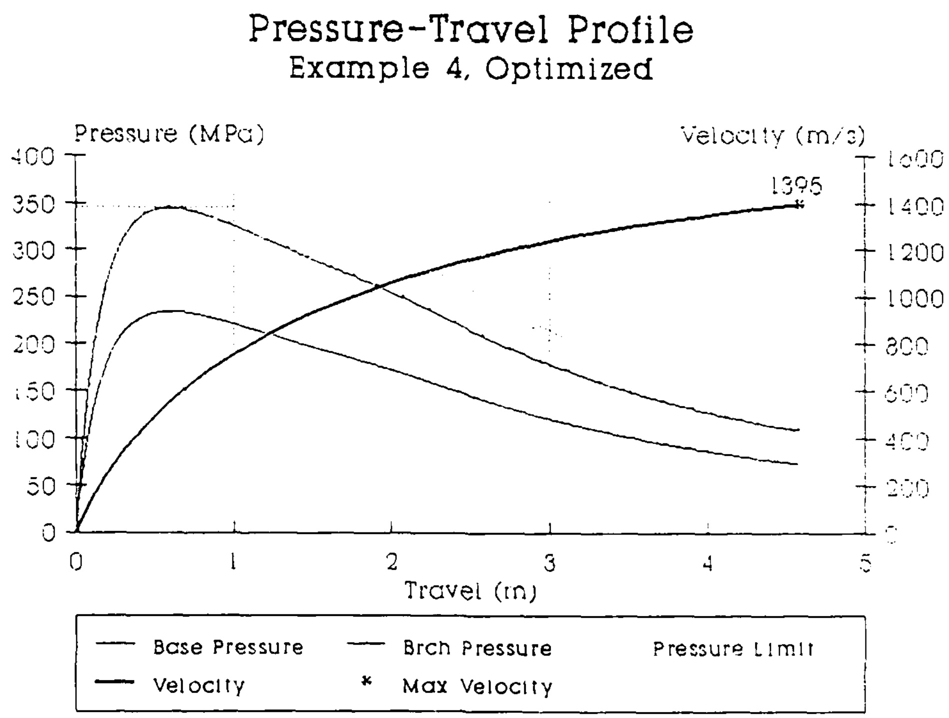


Figure 5.4.4 Example 4 Pressure-Travel Profiles.

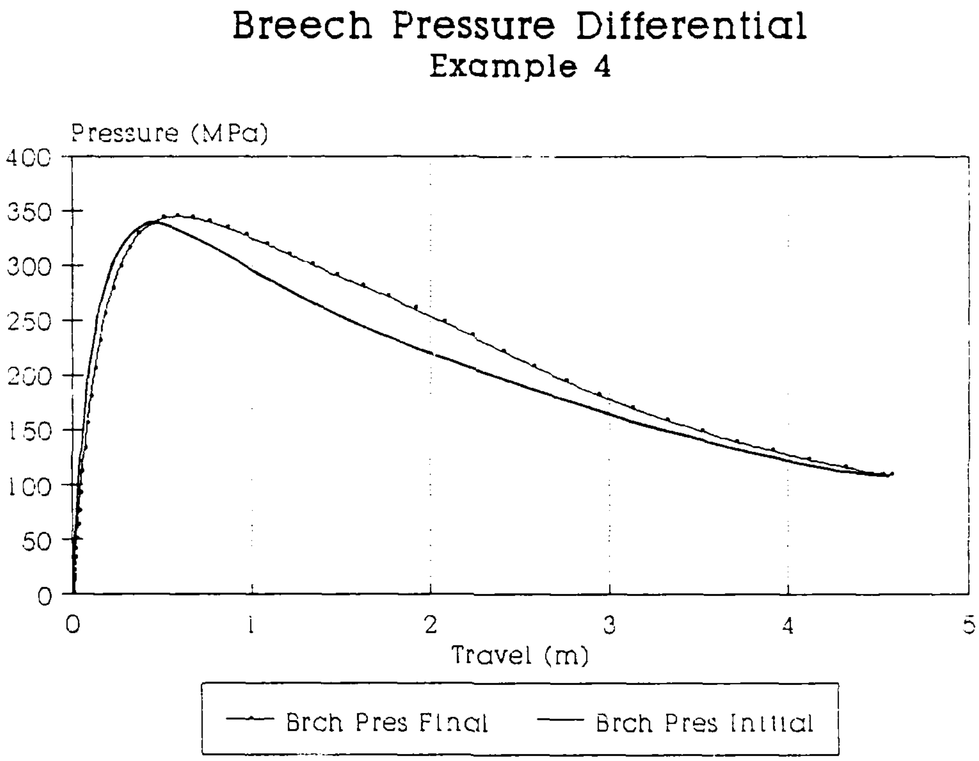


Figure 5.3.5 Example 4 Breech Pressure Differential.

7. Analysis.

The analysis is divided into two parts. The first part is an individual examination of each example problem and the second part is an overall analysis of the trends the example problems indicate.

Both optimizations in Example 1 intialized at the current design and a random point in the parametric space, attain identical projectile performance of 1408 m/s without violating constraints. This indicates that the process is insensitive to the starting points. An analysis of the differences in the grain dimensions between Example 1a and 1b show that the primary difference is in the outer perforation diameter p_o and the grain length L. The relationship between the two values in the region is, for constant velocity, an increase in p_o results in a concurrent decrease in L. This results in similar initial grain surface areas, resulting in comparable projectile velocities. There is also a .7 percent increase in projectile velocity from the current design. This indicates that the process is comparable to current design methods.

In Example 2 the projectile velocity is improved from a non-optimum starting point in the parametric space. Again no constraints are violated. The second optimization improves the projectile velocity from 1397 to 1430 m/s, a difference of 2.7 percent. The optimum is therefore nearly attained in the first optimization. Examination of the one perforation propellent, Propellent 2, shows that the process eliminated the single perforation p_i during optimization. This demonstrates that the process can simplify geometries to improve performance This example demonstrates that the method continues to perform for larger parametric spaces.

The third example resulted in improved projectile performance from a non-optimum point in the parametric space. The velocity, 1368 m/s, is the lowest of the four examples and is a 2.1 percent decrease from Example 1. Since the same propellent thermodynamics are used and identical geometries are present, a closer value could be expected. Analysis of the differences in the problem statements show that the mass constraints prohibited the attainment of the higher velocities reached in Example 1 or 2. When a propellent mass (for a multiple propellent problem) is the active design variable, it is incremented to locate a local optimum. For each change in the active propellent mass, there is no corresponding change for any other propellent mass in the problem. This prohibits a propellent from total elimination by reduction of its mass to zero. Despite this, Example 3 does demonstrate continued performance of the optimization method in a larger parametric space.

Example 4 also improves projectile performance within the constraints. The velocity attained in the first optimization, 1391 m/s is only improved by .30 percent in the second optimization to 1395 m/s. This indicates that in this new parametric space the optimum is nearly attained. The similarity to the previously optimized projectile velocities is due to the relative closeness of the two propellents used, M6 and M8 (see Table 5.1). The question, can the method perform in a different parametric space, is still answered since the propellents are different.

In each example the projectile velocity is improved within the given constraint conditions. Example 1 demonstrates that the scheme will approach the optimum from reasonable starting points and that it matches well with current design techniques. Table 5.2 compares optimized velocities and net improvements.

In Examples 2, 3, and 4 the scheme continues to perform for multiple grains and types of propellents. In Examples 2 and 4 the second optimization provides small improvement indicating that the optimum is nearly attained. This demonstrates the method will locate the optimum in most cases with built in tolerances.

In Example 2 the inner perforation was reduced to zero, showing that the propellent geometry can be simplified if performance is improved. However the mass constraint prohibits elimination of a propellent, through

Example	Inital Velocity	Final Velocity	Percent Change	2nd Itr Change
	m/s	m/s	%	%
1a	1398	1408	+.7	n/a
2	1104	1430	+29.5	+2.4
3	1049	1368	+23.3	n/a
4	1341	1395	+3.9	+.39

Figure 5.2 Performance Synopsis

Example	No. of Function Calls	No. of Variables	Method
1a	573	6	Powell's w/ Golden
2(itr 1)	836	10	Hook-Jeeves
(itr 2)	1024	10	Powell's w/ Golden
3	1394	13	Hook-Jeeves
4(itr 1)	1144	12	Hook-Jeeves
(itr 2)	535	12	Powell's w/ Quadratic

Figure 5.3 Optimization Method Comparison.

the reduction of mass to zero, in multiple propellent problems.

The higher dimensioned parametric space appears to be well behaved as can be seen by the optimization method performance and resultant velocities. Hook-Jeeves and Powell's performed equally, neither showing a distinct advantage. Powell's method used with the three point quadratic approximation (Example 4, Optimzation 2) does converge in less function calls. However, if this method is used near an unfeasible region in the parametric space this advantage will be offset. Table 5.3 compares optimization performance by example.

The questions from Section 6 have been answered by the example problems. Example 1 demonstrated that the method attains comparable performance levels with current design methods. All four of the example problems show that a practical optimum design is attained regardless of the size of the parametric space. Finally, the differences in the four example problems and the relative ease with which the design vector and constraint set can be set show the flexibility and ease of use of the method.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

1. Conclusions.

A general method is developed for optimum and automated propellent grain design of a constrained multivariable interior ballistics system. The results obtained in Chapter V indicate that the method is computationally feasible and yields results comparable with current hunt and search design methods.

This automated design process is an aid to the interior ballistician. It is straightforward to implement and does not require heavy computational support. The interior ballistics model can be quickly changed or improved without affecting the optimization scheme. The process does include constraint effects directly and is flexible in that constraint parameters can be changed without difficulty. The method is tested on a specific problem, but the application is not restricted since the example's characteristics are shared by a large class of interior ballistics problems.

2. Recommendations for Future Research.

The software developed in this research should be tested on several more problems with different parametric spaces. The accuracy of the program can be estimated by solving a variety of actual problems with known solutions. The numerical techniques used in this present work can be improved, both in computational efficiency and convergence speed. A first order method should be integrated into the ALM process to allow greater flexibility.

The integration of both exterior and terminal ballistic models to extend the design capabilities is a long term goal.

REFERENCES

1. Anderson, Ronald D. and Ficke, Kurt D., "IBHVG2 -- A User's Guide", Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, January 1987.

2. Baer, Paul G., "MPRGUN, A Multipurpose-Multipropellent Weapon Interior Ballistic Simulator", Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, 1978.

3. Baer, Paul B. and Frankle, J.M., "The Simulation of Interior Ballistic Performance of Guns by Digital Computer Programs", Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, January 1987.

4. Brent, Richard P., "Algorithms for Minimization without Derivatives", Prentice-Hall Inc., 1973.

5. Burley, D.M., "Studies in Optimization", John Wiley & Sons, 1974.

6. Burnett, J.R. et al., "Projectile Design and Analysis System (PRODAS) Technical Manual", Armament Systems Department, General Electric Corporation, Burlington, Vermont, April 1987.

7. Corner, John A., "Theory of Interior Ballistics of Guns", Jon Wiley & Sons Inc., 1950.

8. Fletcher, R., "Practical Methods of Optimization", John Wiley & Sons, 1980.

9. Fletcher, R., Editor, "Optimization: Symposium of the Institute of Mathmatics and its Applications", Academic Press, 1969.

10. Hiriart-Urruly, Jean B., Editor, "Optimzation Theory and Algorithms", Marcel Dekker Inc, 1983.

11. Krier, Herman and Summerfield, Martin., Editors, "Interior Ballistics of Guns", Progress in Astronautics and Aeronautics, Volume 66, American Institute of Aeronautics and Astronautics, 1979.

12. Lavi, Abrahim and Vogl, Thomas P., Editors, "Recent Advances in Optimzation Techniques", Proceedings of Symposium on Recent Advances in Optimzation Techniques, John Wiley & Sons, 1966.

13. Papalambros, Panos, Y. and Wilde, Douglas, J., "Principles of Optimal Design", Cambridge University Press, 1988.

14. Powell, M.J.D., "A Survey of Numerical Methods for Unconstrained Minimization", pg 43-61, Studies in Optimization 1, Society for Industrial and Applied Mathmatics, 1970.

15. Robbins, Frederick W. and Raab, Thimothy S., "A Lumped Parameter Interior Ballistic Code Using the TTCP Model", Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, November 1988.

16. Stiefel, Ludwig, Editor, "Gun Propulsion Technology", Progress in Astronautics and Aeronautics, Volume 109, American Institute of Aeronautics and Astronautics, 1979.

17. Vanderplaats, Garret N., "Numerical Optimization Techniques for Engineering Design: with Applications", McGraw-Hill Book Company, 1984.

18. Course Notes, "Weapon Systems Engineering", Department of Engineering, United States Military Academy, West Point, New York, 1977.

19. Personnal communication with Mr. Fredrick W. Robbins, Interior Ballistics Division, Ballistics Research Laboratory, Aberdeen Proving Grounds, Maryland, February 1990.

APPENDIX I

OPTIMIZATION CODE

This appendix contains the optimization code used in this thesis. The first item is the declaration file followed by the main progam and the appropriate subroutines. Three called subprograms are not included. The objective function 'fun_int.ftn' is Appendix II. The constraint subroutines 'fun_con.ftn' are included with each example problem. The subroutine that reads the input file 'read_data.ftn', although called by the main program, is included in Appendix II. The files are listed below.

1. declaration.ins.f.
2. optimum.ftn.
3. hook_jeeves.ftn.
4. powell.ftn.
5. search.ftn.
6. ugrid_1d.ftn.
7. gold.ftn.
8. quad.ftn.
9. gaussz.ftn.
10. funx.ftn.
11. check_print.ftn.
12. printit.ftn.
13. tol_test.ftn.
14. update.ftn.

```
C*********************************************************************
C*                      'declarations.ins.f'                         *
C*                                                                   *
C*      DECLARATIONS, PARAMETERIZATION & COMMON BLOCK FILE FOR      *
C*                THE AUGMENTED LAGRANGIAN MULTIPLIER               *
C*                        JOE ROBERT GONZALEZ                        *
C*                           562-88-9645                             *
C*********************************************************************
C*          THIS FILE IS CALLED FROM THE PROGRAM 'optimum.ftn'       *
C*********************************************************************
c  In the common /PARTIALS/ block the following definitions apply:
c     rpa          = the ALM pseudo-objective function rp.
c     rp_max       = the maximum value rp is allowed to attain.
c     lam_h(i)     = the LaGrangian multiplier for the equality
c                       constraints (that is lambda for the H's).
c     lam_g(j)     = the LaGrangian multiplier for the inequality
c                       constraints (that is lambda for the G's).
c     fh(i)        = the calculated value of the respective equality
c                       constraint.
c     fg(j)        = the calculated value of the respective inequality
c                       constraint.

c  In the common /CONTROLS/ block the following definitions apply:
c     calls        = the number of function evaluation calls.
c     ifault       = the 'no local minimum can be found' flag.
c     flag         = a general purpose integer flag.

c  In the main portion of the program the following definitions apply in
c  addition to those listed above:
c     af            = the acceleration factor for hooke-jeeves.
c     converge      = the 'convergance' flag for the ALM.
c     d             = detla, the search interval for hooke jeeves.
c     eps           = epsilon, the tolerance for use in the line searches.
c     fcost         = the current value of the ALM function.
c     fcost_p       = the previous value of the ALM function.
c     fun_cost      = the current value of the objective function.
c     fun_costp     = the previous value of the objective function.
c     gamma         = the gamma in the ALM pseudo-objective function.
c     i,j,k,l,m,n   = working counters.
c     imax          = the maximum array size allocated.
c     itr           = the number of ALM iterations performed.
c     lam_hp(i)     = the previous lamda for the equality constraints.
c     lam_gp(j)     = the previous lamda for the inequality constraints.
c     novar         = the number of independent variables (n).
c     num_h         = the number of equality constraints.
c     p             = a 'p' appended to a variable name is used to
c                     designate the previous value of that variable.
c     technique     = the selection for the line search technique.
c                     1 = Powell's method.
c                     2 = Hook-Jeeves method.
c     tolerance     = the ALM tolerance for use in the main program.
c     xfinal(n)     = the final calculated solution.
c     xinit(n)      = the start point for the optimization.
C*********************************************************************
c  subroutines defined in header of 'opti.f'...........................
      EXTERNAL  FUNX,FUN_FHG,SEARCH,UGRID,GOLD,QUAD,GAUSSZ

      INTEGER*2 I_WANT,I,J,K,L,M,N,NOVAR,NUM_G,NUM_H,FLAG
      INTEGER*2 CALLS,IFAULT,IMAX,ITR,TECHNIQUE

c  parameter establishment............................................
      PARAMETER (IMAX = 30)

      REAL*4  XINIT(IMAX),XFINAL(IMAX),FCOST,FCOST_P,FG(IMAX),FUN_COST
      REAL*4  LAM_H(IMAX),LAM_HP(IMAX),LAM_G(IMAX),LAM_GP(IMAX)
      REAL*4  RPA,RP_MAX,RP_PRIME,GAMMA_PRIME,GAMMA,FH(IMAX),FUN_COSTP
      REAL*4  AF,D,EPS,TOLERANCE

c  common blocks......................................................
      COMMON/PARTIALS/RPA,RP_MAX,LAM_H,LAM_G,FH,FG
      COMMON/CONTROLS/CALLS,IFAULT,FLAG,NUM_H,NUM_G
      COMMON/VALUES   /FUN_COST,FUN_COSTP

      LOGICAL CONVERGE
C*********************************************************************
```

```
C*********************************************************************
C*              THE AUGMENTED LAGRANGIAN MULTIPLIER METHOD           *
C*                     BY:   JOE ROBERT GONZALEZ                     *
C*********************************************************************
C*   TO MINIMIZE FUNCTIONS OF MORE THAN ONE DIMENSION THAT HAVE      *
C*   EQUALITY AND/OR INEQUALITY CONSTRAINTS.                         *
C*                                                                   *
C*    THIS PROGRAM USES FOLLOWING CONSTRAINING ALGORITHM:            *
C*     1)  THE AUGMENTED LAGRANGIAN MULTIPLIER                       *
C*                                                                   *
C*    WITH THE FOLLOWING MINIMIZATION ROUTINES:                      *
C*     1)  POWELL'S METHOD W/ LINE SEARCH                            *
C*     2)  THE METHOD OF HOOK-JEEVES                                 *
C*                                                                   *
C*   TO MINIMIZE UNCONSTRAINED FUNCTIONS THAT HAVE EQUALITY AND/OR   *
C*   INEQUALITY CONSTRAINTS OF MORE THAN ONE DIMENSION.              *
C*                                                                   *
C*********************************************************************
C*   THE SUBROUTINES CALLED FROM USER MAIN ARE (INDENTED NAMES ARE   *
C*   CALLED BY THE PRECEEDING SUBROUTINES):                          *
C*                                                                   *
C*   TOL_TEST    : DETERMINES IF THE ALM TOLERANCE HAS BEEN MET.     *
C*   UPDATE      : UPDATES THE LAGRANGIAN MULTIPLIERS FOR THE PSUEDO-*
C*                    OBJECTIVE FUNCTION.                            *
C*   CHECK_PRINT : PRINTS OUT CURRENT VALUES FOR EACH ALM ITERATION. *
C*   PRINTIT     : PRINTS OUT THE FINAL VALUES.                      *
C*   POWELL      : CONTROLS THE POWELL'S METHOD SEARCH.              *
C*    SEARCH     : CONTROLS THE LINE SEARCH SUBROUTINES.             *
C*     UGRID     : UNIFORM GRID SEARCH METHOD FOR MINIMUM BRACKETING. *
C*     GOLD      : THE GOLDEN SECTIONS METHOD FOR THE MINIMUM.       *
C*     QUAD      : THE QUADRATIC APPROXIMATION METHOD FOR THE MINIMUM. *
C*      GAUSSZ   : SOLUTION TO SYSTEM OF EQUATIONS BY GAUSSIAN       *
C*                    REDUCTION WITH PARTIAL PIVOTING.               *
C*   HOOK_JEEVES : PERFORMS THE HOOK-JEEVES METHOD.                  *
C*   READ_DATA   : THE INPUT DATA SUBROUTINE FILE FOR FUN_INT.FTN    *
C*   FUNX        : PERFORMS THE ALM MODIFICATIONS TO DETERMINE THE   *
C*                    VALUE OF THE PSEUDO-OBJECTIVE FUNCTION.        *
C*    FUN_INT    : PERFORMS THE CALCULATION OF THE COST FUNCTION.    *
C*    FUN_CON    : PERFORMS THE ASSOCIATED EQUALITY AND INEQUALITY   *
C*                    CONSTRAINT FUNCTION EVALUATIONS.               *
C*                                                                   *
C*********************************************************************
      PROGRAM OPTIMUM
%INCLUDE 'declarations.ins.f'
      INTEGER*2 ITYPE

C  INITIALIZATION OF COUNTERS AND FLAGS.................................
      ITR    = 1
      CALLS  = 0
      IFAULT = 0
      VIRGIN = 0
      ITYPE  = 0

      RPA    = 100.0
      RP_MAX = 1.00e8
      GAMMA  = 2.0

      FCOST      = .00
      FUN_COST   = .00
      FCOST_P    = 1.00e-6
      FUN_COSTP  = 1.00e-6

      CONVERGE    = .FALSE.

      TOLERANCE = .2
      EPS       = .0001

C  USER INPUT ..........................................................
      PRINT*,' ENTER NUMBER OF DESIGN VARIABLES:'
      READ*, NOVAR
      PRINT*,' ENTER NUMBER OF EQUALITY/INEQUALITY CONSTRAINTS:'
      READ*,NUM_H,NUM_G

      PRINT*,' NUMBER OF DESIGN VARIABLES    :',novar
      PRINT*,' NUMBER OF EQUALITY CONSTRAINTS:',num_h
      PRINT*,' NUMBER OF EQUALITY CONSTRAINTS:',num_g

C  INITIALIZE Lamda's...................................................
      DO 10 I=1,NUM_H
        LAM_HP(I) = 0.0
10      LAM_H(I)  = 1.0
      DO 20 I=1,NUM_G
        LAM_GP(I) = 0.0
20      LAM_G(I)  = 1.0
```

```
C  READ DATA INPUT SET...................................................
        CALL READ_DATA(XINIT,NOVAR)

C  MINIMIZATION ROUTINE SELECTION........................................
90     PRINT*,'  SELECT MINIMIZATION TECHNIQUE:'
       PRINT*,'      POWELLS METHOD => 1'
       PRINT*,'      HOOKE-JEEVES   => 2'
       READ*, TECHNIQUE




       IF (TECHNIQUE.EQ.1) THEN
         PRINT*,' YOU HAVE SELECTED POWELLS METHOD'
         PRINT*,' ENTER "0" FOR GOLDEN SECTIONS.'
         PRINT*,' ENTER "1" FOR QUADRATIC APPROXIMATION.'
         READ*,ITYPE
       ELSE if (TECHNIQUE.EQ.2) then
         PRINT*,' YOU HAVE SELECTED HOOKE-JEEVES'
         PRINT*,' ENTER SEARCH DELTA [<1] AND ACCELERATION FACTOR [>=1]'
         READ*,D,AF
         PRINT*,' SEARCH DELTA =',D
         PRINT*,' ACCEL FACTOR =',AF
       END IF

C  MINIMIZATION EXECUTION................................................
100    IF (TECHNIQUE.EQ.1) THEN
         CALL POWELL(XINIT,EPS,NOVAR,XFINAL,FCOST,ITYPE)
       ELSE IF (TECHNIQUE.EQ.2) THEN
         CALL HOOK_JEEVES(XINIT,EPS,NOVAR,D,AF,XFINAL,FCOST)
       END IF

C  DIAGNOSTIC PRINT......................................................
       CALL CHECK_PRINT(ITR,XFINAL,FCOST,FCOST_P,NOVAR)

C  CHECKING FOR CONVERGENCE..............................................
       CALL TOL_TEST(LAM_HP,LAM_GP,TOLERANCE,CONVERGE,FCOST_P,FCOST)

C  IF CONVERGENCE FAILED CONTINUE (check alm max itr)....................
       IF ((CONVERGE).EQV.(.FALSE.)) THEN
         IF (ITR.GE.25) THEN
           PRINT*,'    *** MAX ITERATIONS EXCEEDED ***'
           PRINT*,'      THE LAST SET OF VALUES ARE:'
           CALL CHECK_PRINT(ITR,XFINAL,FCOST,FCOST_P,NOVAR)
         ELSE
           CALL UPDATE(LAM_HP,LAM_GP,GAMMA)
           ITR        = ITR + 1
           FCOST_P    = FCOST
           FUN_COSTP = FUN_COST
           DO 101 I = 1,NOVAR
101          XINIT(I) = XFINAL(I)
           GOTO 100
         END IF
       END IF

C  PRINT RESULTS.........................................................
1000   CALL PRINTIT(XFINAL,FCOST,NOVAR)

       STOP
       END
C  END OF USER_MAIN....................................................
```

```
C***********************************************************************
      SUBROUTINE HOOK_JEEVES(IX,EPS,NOVAR,DELTA,ALFA,XI,FXI)
C***********************************************************************
C  This is the Hook-Jeeves unconstrained minimization algorithm.
C***********************************************************************
C  WHERE THE FOLLOWING DEFINTIONS APPLY:
C
C      NOVAR   = NUMBER OF VARIABLES (HEREAFTER REFERED TO AS "N").
C      CALLS   = THE NUMBER OF FUNCTION CALLS.
C      IX(N)   = THE START POINT FOR THE SEARCH.
C      XI(N)   = THE CURRENT X SET.
C      XP(N)   = THE PREVIOUS X SET.
C      YI(N)   = THE CURRENT Y SET.
C      YSP(N)  = THE CURRENT Y SET + DELTA* DJ(N) .
C      YSM(N)  = THE CURRENT Y SET - DELTA* DJ(N) .
C      DJ(N)   = THE UNIT DIRECTION VECTOR FOR VARIABLE N.
C      DELTA   = THE STEP SIZE FOR THE SEARCH.
C      ALFA    = THE ACCELERATION FACTOR FOR THE SEARCH.
C      EPS     = THE "EPSILON" OR TOLERANCE FOR THE SOLUTION.
C      TIMES_THROUGH = THE NUMBER OF SEARCH SETS FOR THE METHOD.
C***********************************************************************
%INCLUDE 'declarations.ins.f'

      REAL*4  IX(IMAX),XI(IMAX),YI(IMAX),DJ(IMAX),YSP(IMAX),YSM(IMAX),
     +        XP(IMAX),FXI,FYSP,FYSM,FYI,TIMES_THROUGH,ALFA,DELTA

      TIMES_THROUGH = 0

C  Assign the working vectors............................................
      DO 100 N=1,NOVAR
        XI(N)=IX(N)
        YI(N)=IX(N)
100   CONTINUE

C  Main loop.............................................................
70    DO 200 N=1,NOVAR
C  Assign the search directions..........................................
        DO 150 M=1,NOVAR
150       DJ(M)=0
        DJ(N)=1

        DO 160 M=1,NOVAR
160       YSP(M)=YI(M)

        YSP(N)=YI(N)+DELTA*DJ(N)
        CALL FUNX(YI,FYI,NOVAR)
        CALL FUNX(YSP,FYSP,NOVAR)

        IF (FYSP.LT.FYI) THEN
          YI(N)=YSP(N)
          FYI=FYSP
        ELSE
          DO 170 M=1,NOVAR
170         YSM(M)=YI(M)
          YSM(N)=YI(N)-DELTA*DJ(N)
          CALL FUNX(YSM,FYSM,NOVAR)
          IF (FYSM.LE.FYI) THEN
            YI(N)=YSM(N)
            FYI=FYSM
          END IF
        END IF
200   CONTINUE

C  Improvement so accelerate........................ .. ...........
      CALL FUNX(XI,FXI,NOVAR)
      IF (FYI.LT.FXI) THEN
        DO 250 N=1,NOVAR
          XP(N)=XI(N)
          XI(N)=YI(N)
          YI(N)=XI(N)+ALFA*(XI(N)-XP(N))
250     CONTINUE
        TIMES_THROUGH = TIMES_THROUGH +1
        GOTO 70

C  Max iteration check...................................................
      ELSE IF (TIMES_THROUGH.GT.50) THEN
        PRINT*,' *** hooke jeeves iterations .GT. 50 ***'
        RETURN

C  End of Minimization...................................................
      ELSE IF (DELTA.LT.EPS) THEN
        RETURN
```

```
      ELSE
         DELTA=DELTA/2
         DO 300 N=1,NOVAR
300        YI(N)=XI(N)
      END IF
      TIMES_THROUGH =TIMES_THROUGH + 1
      GOTO 70
      END

C  END OF HOOK-JEEVES.................................................
```

```
C***********************************************************************
      SUBROUTINE POWELL(IX,EPS,NOVAR,XI,FXI,ITYPE)
C***********************************************************************
C  This subroutine is the alogirthm for Powell's method.  It controls
C  the search directions and the convergence in at this level. The
C  searches are conducted in the following subroutines:
C
C      SEARCH= THE CONTROLLING SEARCH SUBROUTINE.
C      QUAD  = A QUADRATIC APPROXIMATION FOR THE MINIMUMIN LINE SEARCH.
C               (USES 'GAUSSZ' TO SOLVE FOR MINIMUM).
C      GOLD  = THE GOLDEN SECTIONS METHOD FOR THE MINIMUM IN LINE SEARCH.
C      UGRID = UNIFORM GRID SEARCH METHOD FOR MINIMUM BRACKETING.
C***********************************************************************
C  VARIABLE DEFINTIONS:
C      IX(N)         =  INTIAL GUESS OF MINIMUM POINT
C      XI(N)         =  THE CURRENT WORKING X MINIMUM DURING THE SEARCH
C      XN(N)         =  THE HOLDER DURING THE REASSIGNMENT OF THE NEXT XI
C      XP(N)         =  THE PREVIOUS X MINIMUM AS THE SEARCH PROGRESSES
C      YI(N)         =  THE CURRENT WORKING Y SEARCH LOCATION
C      YS(N)         =  THE Y POINT THAT IS "S"ENT INTO THE LINE SEARCH
C      YM(N)         =  THE LINE SEARCH Y "M"INIMUM THAT IS RETURNED
C      DJ(N)         =  THE CURRENT DIRECTION VECTOR OF UNIT LENGTH
C      DJNORM        =  THE NORM OF THE CALCULATED DJ VECTOR
C      LAMXDJ        =  RUNNING SUM OF THE PRODUCT OF THE LAMDA AND THE DJ
C      DJSUM         =  THE RUNNING SUM OF THE NORM CALCULATION
C      EPS           =  THE TOLERANCE
C      XNS           =  X POINTS NORMAL CALCULATION SUM HOLDER
C      H(N,N)        =  THE H MATRIX WHERE THE DIRECTION VECTORS ARE STORED
C      HOLD(N,N+1)   =  THE TRANSITION MATRIX WHERE THE NEW DIRECTION
C                          VECTORS ARE DETERMINED.
C      NORMX         =  THE NORM OF THE XI AND XP VECTOR TO DETERMINE THE
C                          TOLERANCE FIT.
C      LOOKER        =  THE NUMBER OF ITERATIONS OF POWELL'S DONE.
C***********************************************************************
%INCLUDE 'declarations.ins.f'

      INTEGER*2 LOOKER,ITYPE
      REAL*4    XI(IMAX),IX(IMAX),XP(IMAX),DJ(IMAX),H(IMAX,IMAX),
     +    FYM,FXI,FYI,YI(IMAX),LAM(IMAX),HOLD(IMAX,IMAX+1),xn(IMAX),
     +  YM(IMAX),YS(IMAX),NORMX,DJNORM,DJSUM,XNS,LAMXDJ,dnpn(IMAX)

      INTRINSIC SQRT

      LOOKER = 0

      DO 100 N=1,NOVAR
        XI(N) = IX(N)
        YI(N) = IX(N)
100   CONTINUE

C  INITIALIZING THE HOLD MATRIX & PUTTING THE ORIGINAL DIRECTION VECTORS
C  IN THE H MATRIX.....................................................
99    DO 150 K=1,NOVAR
        HOLD(K,NOVAR+1) = 0
        DO 150 M=1,NOVAR
          HOLD(K,M) = 0
          IF (K.EQ.M) THEN
            H(K,M) = 1
          ELSE
            H(K,M) = 0
          END IF
150   CONTINUE

C  GETTING THE CURRENT DJ OUT OF THE H MATRIX FOR THE CURRENT N.........
C  START OF THE MAIN N COUNTING LOOP FOR THE NUMBER OF VARIABLES........
200   DO 300 N=1,NOVAR+1
C  ASSIGNING THE SENDING Y FOR THE SEARCH...............................
        DO 250 M=1,NOVAR
250       YS(M) = YI(M)

c  CURRENT SEARCH DIRECTION.............................................
201     DO 225 M=1,NOVAR
225       DJ(M) = H(M,N)

C  THE SEARCH CALL......................................................
      CALL SEARCH (YS,DJ,EPS,N,FYM,YM,LAM,NOVAR,itype)
```

```
C  A FUNCTION EQUALITY PROBLEM CHECK, IF=1 THEN CAN'T FIND MIN IN SEARCH
C  SO RESET SEARCH DIRECTIONS AND START OVER FROM LAST FEASIBLE POINT.
        IF (IFAULT.EQ.1) THEN
          STOP 'RESTART PROCESS'
        END IF
C  SAVING THE DIRECTION VECTOR FOR THE CURRENT N.......................
        DO 275 M=1,NOVAR
          HOLD(M,N) = LAM(M)*DJ(M)
          H(M,N)    = HOLD(m,n)
          YI(M)     = YM(M)
275     CONTINUE
        FYI = FYM
c  THE N+1 SEARCH DIRECTION............................................
        IF (N.EQ.NOVAR) THEN
          DO 280 M=1,NOVAR
280         H(M,NOVAR+1)=YI(M) - XI(M)

        END IF
300   CONTINUE

C  SETTING THE X'S AND Y'S TO THEIR NEW VALUES.........................
      DO 325 M=1,NOVAR
        XP(M) = XI(M)
        XI(M) = YI(M)
325   CONTINUE
      FXI = FYI
      looker = looker + 1

C  CALCULATING THE NORM OF THE LAST 2 X POINTS TO CHECK CONVERGANCE.....
      XNS=0
      DO 350 M=1,NOVAR
350     XNS = XNS+(XI(M)-XP(M))**2
      NORMX = SQRT(XNS)

C  IF CONVERGANCE HAS BEEN REACHED, XI, FXI IS RETURNED TO MAIN.........
      IF (NORMX.LT.EPS) then
        RETURN
c  iterations exceeding allowable......................................
      ELSE IF (LOOKER.GT.50) then
        PRINT*,'### POWELL ITERATIONS > 50 ###'
        RETURN
      ELSE
C  UPDATING THE SEARCH DIRECTIONS (DJ'S) FOR THE NEXT PASS.............
        DO 410 J=1,NOVAR
          LAMXDJ = 0
          DO 400 M=1,NOVAR
400         LAMXDJ = LAMXDJ+HOLD(J,M)
410       HOLD(J,NOVAR+1) = LAMXDJ

c  adding the directions together.......................................
        DO 420 J=1,NOVAR+1
          DJSUM = 0
          DO 430 K=1,NOVAR
430         DJSUM = DJSUM+HOLD(k,j)**2
          DJNORM = SQRT(DJSUM)
          DO 440 K=1,NOVAR
C  SETTING THE NEW DJ'S TO A UNIT LENGTH TO PREVENT CRAWLING TO A SOLUTION
440         HOLD(k,j) = HOLD(k,j)/DJNORM
420     CONTINUE

C  POSITION OF THE REVISED DJ'S IN THE H MATRIX FOR USE................
        DO 450 n=1,NOVAR
          DO 450 m=1,NOVAR
450         H(M,N) = HOLD(M,N+1)
      END IF
      GOTO 200
      END

C END OF POWELL.......................................................
```

```
C***********************************************************************
      SUBROUTINE SEARCH(YS,DJ,EPS,N,FYMIN,YMIN,LAMNORM,NOVAR,itype)
C***********************************************************************
C  THIS SUBROUTINE CONTROLS THE SEARCH PROCEDURES FOR POWELL' METHOD.
C  WHERE TO FOLLOWING DEFINTIONS APPLY.................................
C    YS      = THE SENT VALUE TO START FROM
C    DIR     = THE DIRECTION OF THE SEARCH
C    DJ      = THE SENT DIRECTION TO SEARCH
C    IFAULT  = THE EQUALITY PROBLEM FLAG
C    ITYPE   = THE LINE SEARCH SELECTION FLAG
C    LAMNORM = THE NORMALIZED DELTA OF THE SEARCH
C    YMIN    = THE RETURNED MINIMUM IN THAT DIRECTION
C    XASTRT  = THE INTERVAL START FROM UGRID
C    XBEND   = THE INTERVAL END FROM UGRID
C    SAV_YS  = THE RETAINED VALUES OF THE START POINT
C***********************************************************************
%INCLUDE 'declarations.ins.f'

      INTEGER*2 ITYPE
      REAL*4    YS(IMAX),DJ(IMAX),YMIN(IMAX),FYMIN,DIR,SAV_YS(IMAX),
     +          LAMNORM(IMAX),LAMRAW,XASTRT(IMAX),XBEND(IMAX)

      INTRINSIC SQRT

      DIR    = 1.0
      IFAULT = 0

C  SAVING THE INCOMING START POINT VALUES..............................
      DO 25 I=1,NOVAR
25      SAV_YS(I)=YS(I)

C CALLING THE UNIFORM GRID SEARCH SUBROUTINE...........................
      CALL UGRID_1d(YS,DJ,NOVAR,XASTRT,XBEND,DIR)
      IF (IFAULT.EQ.1) THEN
        RETURN
      END IF

      IF (ITYPE.EQ.1) THEN
C  CALLING THE QUADRATIC APPROXIMATION INTERVAL REDUCER................
        CALL QUAD(XASTRT,XBEND,YMIN,FYMIN,NOVAR)
      ELSE
C  CALLING THE GOLDEN SECTIONS INTERVAL REDUCER........................
        CALL GOLD(XASTRT,XBEND,EPS,FYMIN,YMIN,NOVAR)
      END IF

C CALCULATING THE FINAL LAMBDA FOR POWELLS METHOD TO RETURN TO POWELL..
      LAMRAW=0
      DO 100 I=1,NOVAR
100     LAMRAW=LAMRAW+(YMIN(I)-SAV_YS(I))**2
      LAMNORM(N)=SQRT(LAMRAW)*DIR

      RETURN
      END

C END OF SEARCH........................................................
```

```
c***********************************************************************
      SUBROUTINE UGRID_1D(XS,DJ,NOVAR,XP,XN,DIR)
c***********************************************************************
C  This subroutine performs a 1 dimensional uniform step search until
C  a interval is found that contains a minimum is found.
c***********************************************************************
C  WHERE THE FOLLOWING DEFINITIONS APPLY:
C
C     XS(N) = THE CURRENT X VALUE
C     XP(N) = THE PREVIOUS X VALUE, INTERVAL START
C     XN(N) = THE NEXT X VALUE, INTERVAL END
C     DJ    = THE DIRECTION VECTOR FOR THE SEARCH
C     DEL   = THE INCREMENT OF THE SEARCH
C     FXS,
c     FXN,
c     FXS   = THE FUNCTION VALUES FOR EACH RESPECTIVE X
C     DIR   = IF THE SEARCH IS NEGATIVE THIS SETS A NEGATIVE DISTANCE
c     COUNT = THE NUMBER OF STEPS TAKEN TO FIND A MINIMUM
C**********************************************************************
%INCLUDE 'declarations.ins.f'

      INTEGER*2 count
      REAL*4    DJ(IMAX),XS(IMAX),XN(IMAX),XP(IMAX),DEL,FXS,FXN,FXP,DIR

      DEL   = 0.001
      count = 0

C  ADDING AND SUBTRACTING THE DEL TO THE INITIAL VALUE................
      DO 100 N=1,NOVAR
        XN(N) = XS(N)+DEL*DJ(N)
        XP(N) = XS(N)-DEL*DJ(N)
100   CONTINUE

      CALL FUNX(XS,FXS,NOVAR)
      CALL FUNX(XN,FXN,NOVAR)
      CALL FUNX(XP,FXP,NOVAR)

c  CASES LISTED OUT...................................................
c  case 1. \/
      IF (FXP.GT.FXS.and.FXN.gt.FXS) THEN
         RETURN
c  case 2. /\
      ELSE IF (FXP.LT.FXS.and.FXN.lt.FXS) THEN
         GOTO 300
c  case 3. -\
      ELSE IF (FXP.EQ.FXS.AND.FXN.lt.FXS) THEN
         GOTO 300
c  case 4. \_
      else if (FXP.GT.fxs.AND.FXN.eq.FXS) then
         goto 300
c  case 5. \\
      else if (FXP.GT.fxs.AND.FXN.lt.FXS) then
         goto 300
c  case 6. //
      else if (FXP.LT.fxs.AND.FXN.gt.FXS) then
         goto 290
c  case 7. _/
      else if (FXP.EQ.fxs.AND.FXN.gt.FXS) then
         goto 290
c  case 8. /-
      else if (FXP.LT.fxs.AND.FXN.eq.FXS) then
         goto 290
c  case 9. --
      ELSE IF (FXP.EQ.FXS.AND.FXN.EQ.FXS) THEN
         goto 390
      END IF

C  IF THE SEARCH IS TO LEFT THEN THIS RESETS THE VALUES TO ALLOW IT....
290   DO 295 N=1,NOVAR
295     XN(N)=XP(N)

      DEL = -DEL
      FXN =  FXP
      DIR = -1.0
```

```
C  THIS IS THE MAIN SEARCH STEP LOOP..................................
300   DO 310 N=1,NOVAR
        XP(N) = XS(N)
        XS(N) = XN(N)
310     XN(N) = XN(N)+DEL*DJ(N)
      FXS=FXN
      CALL FUNX(XN,FXN,NOVAR)

      IF (FXN.LT.FXS) THEN
C  STEP CHECK..........................................................
        IF (COUNT.EQ.100) THEN
          PRINT*,' no minimum found in 100 steps'
          RETURN
        END IF
        COUNT = COUNT +1
        GOTO 300

C  THE FUNCTION VALUES ARE EQUAL AND NO MINIMUM IS FOUND, IFAULT IS SET

      ELSE IF (FXN.EQ.FXS) THEN
390     PRINT*, 'THERE MAY BE AN EQUALITY PROBLEM IN THIS EQUATION.'
        IFAULT=1
      END IF

C  AN INTERVAL HAS BEEN FOUND..........................................
      RETURN
      END

C END OF UNIFORM GRID..................................................
```

```
C*********************************************************************
      SUBROUTINE GOLD(AX,BX,TOL,FXMIN,XMIN,NOVAR)
C*********************************************************************
C  THIS SUBROUTINE PERFORMS A GOLDEN SECTIONS SEARCH FOR THE LOCAL
C  MINIMA IN A GIVEN INTERVAL
C.....................................................................
C  WHERE THE FOLLOWING DEFINITIONS APPLY
C     AX     =  THE START OF THE SEARCH INTERVAL
C     AX     =  END OF THE SEARCH INTERVAL
C     XMIN   =  THE RETURNED MINIMUM VALUE FOR X
C     MU     =  THE GOLDEN SECTIONS MU OF .618 OF INTERVAL
C     LAMDA  =  THE GOLDEN SECTIONS LAMDA OF .382 OF INTERVAL
C     TOL    =  THE TOLERANCE OF THE SOLUTION (COMPARED TO ABNORM)
C     ABNORM =  IS THE LINEAR DISTANCE BETWEEN THE ENDS OF THE INTERVAL
C     MIN    =  THE END OF INTERVAL LOWEST VALUE VARIABLES
C     F      =  THE F PREFIX INDICATES A FUNCTION VALUE
C*********************************************************************
%INCLUDE 'declarations.ins.f'

      REAL*4  AX(IMAX),BX(IMAX),LAMDA(IMAX),XMIN(IMAX),MU(IMAX),AB,
     +        ABNORM,TOL,FA,FB,FMU,FLAMDA,FXMIN,MIN1(IMAX),
     +        MIN2(IMAX),FMIN1,FMIN2

      INTRINSIC SQRT

C  CALCULATION OF THE FIRST MU AND LAMDA...............................
      DO 100 N=1,NOVAR
        MU(N)     = AX(N)+.618*(BX(N)-AX(N))
        LAMDA(N) = AX(N)+.382*(BX(N)-AX(N))
100   CONTINUE

C  THE INITIAL FUNCTION CALLS..........................................
      CALL FUNX(AX,FA,NOVAR)
      CALL FUNX(BX,FB,NOVAR)
      CALL FUNX(MU,FMU,NOVAR)
      CALL FUNX(LAMDA,FLAMDA,NOVAR)

C  THE START OF THE INTERVAL CHECK WITH A DETERMINATION OF ABNORM......
110   AB=0
      DO 125 N=1,NOVAR
125     AB=AB+(BX(N)-AX(N))**2
      ABNORM=SQRT(AB)

200   IF (ABNORM.LE.TOL) THEN
        GOTO 400
      ELSE IF (FLAMDA.LT.FMU) THEN
C  THE INTERVAL IS CONVERGING TO THE LEFT..............................
300     DO 350 N=1,NOVAR
          BX(N) = MU(N)
          MU(N) = LAMDA(N)
          LAMDA(N) = AX(N)+.382*(BX(N)-AX(N))
350     CONTINUE
        FB  = FMU
        FMU = FLAMDA
        CALL FUNX(LAMDA,FLAMDA,NOVAR)
        GOTO 110

      ELSE
C  THE INTERVAL IS CONVERGING TO THE RIGHT.............................
         DO 250 N=1,NOVAR
           AX(N)     = LAMDA(N)
           LAMDA(N) = MU(N)
           MU(N)     = AX(N)+.618*(BX(N)-AX(N))
250      CONTINUE
         FA     = FLAMDA
         FLAMDA = FMU
         CALL FUNX(MU,FMU,NOVAR)
         GOTO 110
      END IF
```

```
C  ONCE THE TOLERANCE IS MET THE LOWEST VALUE IS USED..................
400   IF (FA.LT.FLAMDA) THEN
        do 405 i=1,novar
405       min1(i) = ax(i)
          fmin1   = fa
      ELSE
        do 410 i=1,novar
410       MIN1(i) = LAMDA(i)
          FMIN1   = FLAMDA
      END IF

      IF (FB.LT.FMU) THEN
        do 415 i=1,novar
415       MIN2(i) = Bx(i)
          FMIN2   = fB
      ELSE
        do 420 i=1,novar
420       MIN2(i) = MU(i)
          FMIN2   = FMU
      END IF

      IF (FMIN1.LT.FMIN2) THEN
        do 425 i=1,novar

425       XMIN(i) = MIN1(i)
      ELSE
        do 430 i=1,novar
430       XMIN(i) = MIN2(i)
      END IF
      CALL FUNX(XMIN,FXMIN,NOVAR)

      RETURN
      END

C END OF GOLDEN SECTIONS..............................................
```

```
C***********************************************************************
      SUBROUTINE QUAD(AX,BX,XQMIN,FXQMIN,NOVAR)
C***********************************************************************
C  THIS SUBROUTINE PERFORMS THE QUADRATIC APPROXIMATION OF THE LINE
C  SEARCH MINIMUM.  IT CALLS GAUSSZ.FTN TO SOLVE THE SYSTEM OF
C  EQAUTIONS.
C   NOTE:  SEE GOLD FOR ALL OTHER VARIABLE NAMES
C   ORIG = THE MATRIX OF THE SYSTEMS OF EQUATION GENERATED FOR
C          EACH DIRECTION TO ESTIMATE THE MINIMUM OF X(N)
C   CX   = THE MIDPOINT OF THE AX - BX INTERVAL
C   YY   = THE RETURNED SOLUTION VECTOR FOR EACH ORIG
C***********************************************************************
%INCLUDE 'declarations.ins.f'

      REAL*4  AX(IMAX),BX(IMAX),CX(IMAX),XQMIN(IMAX),FXQMIN,ORIG(3,4)
      REAL*4  YY(3),FA,FB,FC

C  DETERMINING THE VALUE OF C...........................................
      DO 100 N=1,NOVAR
100     CX(N)=(AX(N)+BX(N))/2

      CALL FUNX(AX,FA,NOVAR)
      CALL FUNX(BX,FB,NOVAR)
      CALL FUNX(CX,FC,NOVAR)

C  MAKING EACH ORIG MATRIX, N NUMBER OF TIMES..........................
      DO 200 N=1,NOVAR
        ORIG(1,1)=AX(N)*AX(N)
        ORIG(2,1)=BX(N)*BX(N)
        ORIG(3,1)=CX(N)*CX(N)

        ORIG(1,2)=AX(N)
        ORIG(2,2)=BX(N)
        ORIG(3,2)=CX(N)

        DO 150 I=1,3
150       ORIG(I,3)=1.0

        ORIG(1,4)=FA
        ORIG(2,4)=FB
        ORIG(3,4)=FC

C  THE EQUALITY CHECK IN ANY N DIRECTION FOR A DEFAULT XMIN(N).........
        IF (AX(N).EQ.BX(N)) THEN
          XQMIN(N)=AX(N)
        ELSE

C  THE CALL TO THE MATRIX SOLUTION SUBROUTINE TO SOLVE ORIG............
          CALL GAUSSZ(ORIG,YY,3,4)
          XQMIN(N)=-YY(2)/(2*YY(1))
        END IF
200   CONTINUE
      CALL FUNX(XQMIN,FXQMIN,NOVAR)

      RETURN
      END

C END OF QUAD.........................................................
```

```
C***********************************************************************
      SUBROUTINE FUNX(X,F_ALM,NOVAR)
C***********************************************************************
C   This subroutine converts the objective function to the pseudo-
C  ojective function of the ALM method.
C***********************************************************************
%INCLUDE 'declarations.ins.f'

      REAL*4  X(IMAX),F_ALM,SUMFLH,SUMFH2,SUM_PSI,SUM_PSI2,PSI(IMAX)
      REAL*4  SUM_FH2,SUM_FGT,FGT(IMAX)

      INTRINSIC MAX

      flag  = 0
      calls = calls + 1

C  CALLING THE COST FUNCTION AND CONSTRAINT FUNCTION EVALUATIONS........
      CALL FUN_INT(X,NOVAR,FCOST,virgin,flag)
      CALL FUN_CON(X,NOVAR)

C  ALM TERM GENERATION..................................................
      SUMFLH = 0
      SUMFH2 = 0
      DO 100 I=1,NUM_H
        SUMFLH = SUMFLH+LAM_H(I)*FH(I)
100     SUMFH2 = SUMFH2+FH(I)**2

      DO 110 I=1,NUM_G
110     PSI(I) = MAX(FG(I),-LAM_G(I)/(2*RPA))

      SUM_PSI  = 0
      SUM_PSI2 = 0
      DO 130 I=1,NUM_G
        SUM_PSI  = SUM_PSI+LAM_G(I)*PSI(I)
130     SUM_PSI2 = SUM_PSI2+RPA*PSI(I)**2

c  THE PSEUDO-OBJECTIVE FUNCTION.......................................
      F_ALM = FCOST+SUM_PSI+SUM_PSI2+SUMFLH+RPA*SUMFh2
      FUN_COST = FCOST

c  MAX ITERATION FOR FUNCTION CALLS CHECK...............................
      IF (CALLS.EQ..2000) THEN
        STOP '```````max iterations for method reached```````'
      END IF f

      RETURN
      END

C  END OF ALM FUN.......................................................
```

```
C*********************************************************************
      SUBROUTINE GAUSSZ(A,X,N,N1)
C*********************************************************************
C  THIS SUBROUTINE PERFORMS GAUSSIAN REDUCTION TO SOLVE THE SYSTEM OF
C  EQUATIONS.
C.....................................................................
      INTEGER*2 I,J,K,L,M,N,J1,JJ,N1
      REAL*4    AIJ,X(N),A(N,N1),FA,FB,FC,BIG,DUMMY

      DO 100 J=1,N
        AIJ=A(J,J)
        J1=J+1
        IF (J1.GT.N) GO TO 980
        BIG=ABS(A(J,J))
        M=J
        DO 900 L=J1,N
          IF (ABS(A(L,J)).LE.BIG) GOTO 900
900     CONTINUE

        DO 990 JJ=J,N1
          DUMMY   = A(M,JJ)
          A(M,JJ) = A(J,JJ)
          A(J,JJ) = DUMMY
990     CONTINUE
980     CONTINUE

        DO 200 K=J,N1
200       A(J,K)=A(J,K)/AIJ

        DO 300 I=1,N
          IF (I.EQ.J) GO TO 300
          AIJ=A(I,J)
          DO 400 K=J,N1
400         A(I,K)=A(I,K)-AIJ*A(J,K)
300     CONTINUE
100   CONTINUE

      DO 500 I=1,N
500     X(I)=A(I,N1)

      RETURN
      END

C END OF GAUSSZ.......................................................
```

```
c*********************************************************************
      SUBROUTINE CHECK_PRINT(ITR,XFINAL,FCOST,FCOST_P,NOVAR)
c*********************************************************************
c  This subroutine prints the design vector, function value and pseudo-
c  objective function value for each complete ALM iteration.
c*********************************************************************
%INCLUDE 'declarations.ins.f'

      PRINT*,' AT ITERATION NUMBER',ITR,' AND CALL NUMBER',CALLS
      PRINT*,' THE VALUES OF THE VARIABLES ARE:'

c  Print the current design vector...................................
      do 7 I=1,novar
7       WRITE(*,10)I,xfinal(I)
10    FORMAT(5X,' X(',I2,') = ',1F10.6)
      WRITE(*,11)
11    FORMAT(/)

      IF (FLAG.EQ.1) PRINT*,' *** NOTE: ALL PROPELLENT EXPENDED  ***'

c  Print the current and last function and ALM values.................
      PRINT*,' CURRENT  FCOST = ',fun_cost
      PRINT*,' PREVIOUS FCOST = ',fun_costp
      PRINT*,' CURRENT  ALM   = ',fcost
      PRINT*,' PREVIOUS ALM   = ',fcost_p

      WRITE(*,11)
      PRINT*,' RP             = ',rpa

c  Print the current lambda and value of the constraint function.......
      do 20 I=1,NUM_H
20      WRITE(*,60)I,fh(i),i,lam_h(i)
      do 30 I=1,NUM_G
30      WRITE(*,50)I,fg(i),i,lam_g(i)
50    format(1x,'FG(',I2,') = ',e18.6,' & LAM_G(',I2,') =',e18.6)
60    format(1x,'FH(',I2,') = ',e18.6,' & LAM_H(',I2,') =',e18.6)
      WRITE(*,11)

      RETURN
      END
C  END OF CHECK PRINT................................................



c*********************************************************************
      SUBROUTINE PRINTIT(XFINAL,FCOST,NOVAR)
c*********************************************************************
c  This subroutine prints the final design vector, function value and
c  number of function calls.
c*********************************************************************
%INCLUDE 'declarations.ins.f'

      PRINT*, ' THE FINAL FUNCTION VALUE IS (m/s):', FUN_COST*(-1.0)
      PRINT*, ' THE ',NOVAR,' VARIABLE VALUES ARE:'
      DO 7 I=1,novar
7       WRITE(*,10)I,xfinal(I)
10    FORMAT(5X,' X(',I2,') = ',1F10.6)
      WRITE(*,11)
11    FORMAT(/)

      WRITE(*,20)((xfinal(I)*100),I=1,NOVAR)
20    FORMAT(10X,8F12.5)
      PRINT*, ' THE TOTAL NUMBER OF FUNCTION CALLS WAS :', CALLS
      PRINT*, ' THE FINAL ALM FUNCTION VALUE WAS       :', FCOST

      RETURN
      END

C  END OF PRINT IT..................................................
```

```
C***********************************************************************
      SUBROUTINE TOL_TEST(LAM_HP,LAM_GP,TOLERANCE,CONVERGE,FCOST_P,
     +                    FCOST)
C***********************************************************************
C  This subroutine determines if the convergence has occured for the
C  ALM method to terminate.
C  LOCAL DEFINTIONS.................................................
C      TOTAL   = THE SUM OF THE NUMBER OF INEQUALITY AND EQUALITY
C                CONSTRAINTS.
C      DIFFH   = THE DIFFERENCE BETWEEN THE CURRENT AND PREVIOUS LAMBDA
C                FOR EQUALITY CONSTRAINTS.
C      DIFFG   = SEE ABOVE, FOR INEQUALITY CONTRAINTS.
C      ALM_SUM = THE COUNTER FOR THE NUMBER OF CONSTRAINTS IN TOLERANCE.
C      DELTA_C = THE CHANGE IN THE COST FUNCTION SINCE LAST ITERATION.
C.......................................................................
%INCLUDE 'declarations.ins.f'

      INTEGER*2 TOTAL,ALM_SUM
      REAL*4    DIFFH,DIFFG,DELTA_C

      INTRINSIC SQRT

      ALM_SUM = 0
      DELTA_C = ABS(FUN_COST - FUN_COSTP)
      TOTAL   = NUM_H + NUM_G

      DO 100 I=1,NUM_H
        DIFFH = LAM_H(I)-LAM_HP(I)
        IF (DIFFH.LE.TOLERANCE) THEN
          ALM_SUM = ALM_SUM+1
        END IF
100   CONTINUE

      DO 110 I=1,NUM_G
        DIFFG = LAM_G(I)-LAM_GP(I)
        IF (DIFFG.LE.TOLERANCE) THEN
          ALM_SUM = ALM_SUM+1
        END IF
110   CONTINUE

      IF (DELTA_C.LE.TOLERANCE.AND.ALM_SUM.EQ.TOTAL) THEN
        CONVERGE = .TRUE.
      ELSE
        CONVERGE = .FALSE.
      END IF

      RETURN
      END

C  END OF TOL TEST.....................................................


C***********************************************************************
      SUBROUTINE UPDATE(LAM_HP,LAM_GP,GAMMA)
C***********************************************************************
C  This subroutine updates the lambda's and the rp for the ALM method.
%INCLUDE 'declarations.ins.f'

      INTRINSIC MAX

c  alm update of lambda's for equality constraints.....................
      DO 200 I=1,NUM_H
        LAM_HP(I)=LAM_H(I)
200     LAM_H(I)=LAM_H(I)+2*RPA*FH(I)

c  alm update of lambda's for inequality constraints...................
      DO 210 I=1,NUM_G
        LAM_GP(I)=LAM_G(I)
210     LAM_G(I)=LAM_G(I)+2*RPA*MAX(FG(I),-LAM_G(I)/(2*RPA))

c rp update............................................................
      RPA=GAMMA*RPA
      IF (RPA.GE.RP_MAX) THEN
        RPA=RP_MAX
      END IF

      RETURN
      END

C  END OF UPDATE.......................................................
```

APPENDIX II

INTERIOR BALLISTICS CODE

This appendix contains the interior ballistics code used in this thesis. This is a modified version of IBRGAC (15) to fit the optimization model. The first item is the declaration file followed by the main progam. It is organized by subroutine and includes all of the design vector assignment and return subroutines. The files listed below.

1. intball.ins.f.
2. fun_int.ftn.
3. prf017.ftn.
4. read_data.ftn.
5. reset_data.ftn.
6. mass_check.ftn.
7. var_in.ftn (Example 1-4).
8. var_out.ftn (Example 1-4).

```
C*****************************************************************************
C*                              'intball.ins.f'                              *
C*                                                                           *
C*          DECLARATIONS, PARAMETERIZATION & COMMON BLOCK FILE FOR           *
C*                THE INTERIOR BALLISTICS CODE, IBRGAC (FROM BRL)            *
C*****************************************************************************
C*                                                                           *
C*     THIS FILE IS CALLED FROM THE SUBROUTINE 'fun_int' IN 'optimum.f'      *
C*                                                                           *
C*****************************************************************************
c   The following defintions apply (The s_ prefix indicates the saved original
C   of the variable):

c       Variable  Type:      Variable                              Units:
c       Name:                Meaning:
c   Record 1.....................................................................
c       CHAM      REAL*4     CHAMBER VOLUME                        cm^3
c       GRVE      REAL*4     GROOVE DIAMETER                       cm
c       ALAND     REAL*4     LAND DIAMETER                         cm
c       GLR       REAL*4     GROOVE/LAND RATIO                     none
c       TWST      REAL*4     TWIST                              turns/caliber
c       TRAVP     REAL*4     PROJECTILE TRAVEL                     cm
c       IGRAD     INTEGER*2  GRADIENT FLAG                         none
c                                1 = Lagrange
c                                2 = Chambrage

c   Record 1a....................................................................
c       NCHPTS    INTEGER*2  NUMBER POINTS TO DESCRIBE CHAMBER     none
c   For I=1,nchpts
c       CHDIST(I) REAL*4     INITIAL DISTANCE FROM BREECH          cm
c       CHDIAM(I) REAL*4     DIAMETER AT CHDIST(I)                 cm

c   Record 2.....................................................................
c       PWRT      REAL*4     PROJECTILE MASS                       kg
c       IAIR      INTEGER*2  CALCULATE ENERGY LOST TO AIR
c                              RESISTANCE FLAG                     none
c       HTFR      REAL*4     FRACTION OF WORK DONE AGAINST
c                              BORE TO HEAT TUBE                   none
c       PGAS      REAL*4     GAS PRESSURE IN FRONT OF PROJECTILE   Pa

c   Record 3.....................................................................
c       NPTS      INTEGER*2  NUMBER OF BARREL RESISTANCE POINTS    none
c   For I=1,npts
c       BR(I)     REAL*4     BORE RESISTANCE                       MPa
c       TRAV(I)   REAL*4     TRAVEL                                cm

c   Record 4.....................................................................
c       RCWT      REAL*4     MASS OF RECIOLING PARTS               kg
c       NRP       INTEGER*2  NUMBER OF RECOIL PAIR POINTS          none
c   For I=1,nrp
c       RP(I)     REAL*4     RECOIL FORCE                          N
c       TR(I)     REAL*4     RECOIL TIME                           s

c   Record 5.....................................................................
c       HO        REAL*4     FREE CONVECTION HEAT TRANSFER
c                              COEFFICIENT                         w/cm^2-k
c       TSHL      REAL*4     CHAMBER WALL THICKNESS                cm
c       CSHL      REAL*4     HEAT CAPACITY OF STEEL OF CHAMBER WALL J/g-k
c       TWAL      REAL*4     INITIAL TEMPERATURE OF CHAMBER WALL   k
c       HL        REAL*4     HEAT LOSS COEFFICIENT                 none
c       RCHOS     REAL*4     DENSITY OF CHAMBER WALL STEEL         g/cm^3

c   Record 6.....................................................................
c       FORCIG    REAL*4     IMPETUS OF IGNITER PROPELLENT         J/g
c       COVI      REAL*4     COVOLUME OF IGNITER                   cm^3/g
c       TEMPI     REAL*4     ADIABATIC FLAME TEMP OF IGNITER       k
c       CHWI      REAL*4     INITIAL MASS OF IGNITER               kg
c       GAMAI     REAL*4     RATIO OF SPECIFIC HEAT FOR IGNITER    none
```

```
c   Record 7...................................................................
c       NPROP     INTEGER*2   NUMBER OF PROPELLENT TYPES                 none
c   For I=1,nprop
c       FORCP(I)  REAL*4      IMPETUS OF PROPELLENT                      J/g
c       TEMPP(I)  REAL*4      ADIABATIC TEMPERATURE OF PROPELLENT        k
c       COVP(I)   REAL*4      COVOLUME OF PROPELLENT                     cm^3/g
c       CHWP(I)   REAL*4      INITIAL MASS OF PROPELLENT                 kg
c       RHOP(I)   REAL*4      DENSITY OF PROPELLENT                      g/cm^3
c       GAMAP(I)  REAL*4      RATIO OF SPECIFIC HEATS FOR PROPELLENT     none
c       NPERFS(I) INTEGER*2   NUMBER OF PERFORATIONS ON PROPELLENT       none
c       GLENP(I)  REAL*4      LENGTH OF PROPELLENT GRAIN                 cm
c       PDPI(I)   REAL*4      DIAMETER OF INNER PERFORATIONS
c                               IN PROPELLENT GRAINS                     cm
c       PDPO(I)   REAL*4      DIAMETER OF OUTER PERFORATIONS
c                               IN PROPELLENT GRAINS                     cm
c       GDIAP(I)  REAL*4      OUTSIDE DIAMETER OF PROPELLENT GRAIN       cm
c       DBPCP(I)  REAL*4      DISTANCE BETWEEN PERFORATION CENTERS       cm

c   Record 8...................................................................
c   For J=1,nprop
c       NBR(J)    INTEGER*2   NUMBER OF BURNING POINTS                   none
c   For I=1,nbr(j)
c       ALPHA(J,I) REAL*4     EXPONENT                                   none
c       BETA(J,I)  REAL*4     COEFFICIENT                     cm/s-MPa^(aplha(j))

c       PRESS(J,I) REAL*4     PRESSURE                                   MPa

c   Record 9...................................................................
c       DELTAT     REAL*4     TIME INCREMENT (STEP)                      ms
c       DELTAP     REAL*4     PRINT INCREMENT                            ms
c       TSTOP      REAL*4     STOP TIME FOR CALCULATIONS                 ms
c
c******************************************************************************
c  parameter defintions................................................
      INTEGER*2 IMAX
      REAL*4    PI

      PARAMETER(IMAX = 20)
      PARAMETER(PI   = 3.14159)

c  File Input/Output...................................................
      CHARACTER*10 bdfile,outfil

c  interior ballistics definitions.....................................
c  Record 1............................................................
      REAL*4    cham,grve,aland,glr,twst,travp
      INTEGER*2 igrad
      REAL*4    s_cham,s_grve,s_aland,s_glr,s_twst,s_travp
      INTEGER*2 s_igrad

c  Record 1a...........................................................
      REAL*4    chdist(10),chdiam(10)
      INTEGER*2 nchpts
      REAL*4    s_chdist(10),s_chdiam(10)
      INTEGER*2 s_nchpts

c  Record 2............................................................
      REAL*4    prwt,htfr,pgas
      INTEGER*2 iair
      REAL*4    s_prwt,s_htfr,s_pgas
      INTEGER*2 s_iair

c  Record 3............................................................
      REAL*4    br(10),trav(10)
      INTEGER*2 npts
      REAL*4    s_br(10),s_trav(10)
      INTEGER*2 s_npts

c  Record 4............................................................
      REAL*4    rcwt,rp(10),tr(10)
      INTEGER*2 nrp
      REAL*4    s_rcwt,s_rp(10),s_tr(10)
      INTEGER*2 s_nrp

c  Record 5............................................................
      REAL*4    ho,tshl,cshl,twal,hl,rhocs
      REAL*4    s_ho,s_tshl,s_cshl,s_twal,s_hl,s_rhocs

c  Record 6............................................................
      REAL*4    forcig,covi,tempi,chwi,gamai
      REAL*4    s_forcig,s_covi,s_tempi,s_chwi,s_gamai
```

```
c Record 7.............................................................
      REAL*4     forcp(10),tempp(10),covp(10),chwp(10),rhop(10),
     +           gamap(10),glenp(10),pdpi(10),pdpo(10),gdiap(10),
     +           dbpcp(10)
      INTEGER*2 nprop,nperfs(10)
      REAL*4     s_forcp(10),s_tempp(10),s_covp(10),s_chwp(10),
     +           s_rhop(10),s_gamap(10),s_glenp(10),s_pdpi(10),
     +           s_pdpo(10),s_gdiap(10),s_dbpcp(10)
      INTEGER*2 s_nprop,s_nperfs(10)

c Record 8.............................................................
      REAL*4     alpha(10,10),beta(10,10),pres(10,10)
      INTEGER*2 nbr(10)
      REAL*4     s_alpha(10,10),s_beta(10,10),s_pres(10,10)
      INTEGER*2 s_nbr(10)

c Record 9.............................................................
      REAL*4     deltat,deltap,tstop
      REAL*4     s_deltat,s_deltap,s_tstop

c end of record declarations.............................................

c Lagrange chamber volume values.........................................
      REAL*4     bore,b1,b2,b3,b4,zz,bint(4),bvol,r1,r2,diam,area,temp,
     +           chmlen
      REAL*4     s_bint(4),s_bvol,s_chmlen,s_bore

c local use declarations.................................................
      REAL*4     step,tmpi,lambda,pmaxm,pmaxba,tpmaxm,tpmaxbr,pmaxbr,
     +           tpmaxba,tpmax,as(4),bs(4),ak(4),vp0,tr0,tcw,ibo(10),
     +           volgi,pmean,volg,wallt,ptime,z(20),y(20)

      REAL*4     dpmaxba,dpmaxbr,max_mass,l_d,p_f,m_r,total_vol_prop,
     +           cham_vol

      INTEGER*2 ibrp,nde,isw1,i1,i,j,k,l,m,y_axis,x_axis

      REAL*4     velocity, values(20,20)

      REAL*4     resp,elpt,elpr,pt,vzp,j4zp,elgpm,elbr,elrc,areaw,avcp,
     +           avc,avden,z18,z19,avvel,htns,elht,air,elar,rfor,areab,
     +           eprop,rprop,tenergy,tgas,v1,cov1,pbase,pbrch,j1zp,
     +           j2zp,j3zp,a2t,alf,a1t,bt,bata,gamma,delta,ds(20),p(20),
     +           t,rmvelo,tmvelo,disto,dfract,efi,efp,tenerg,tengas,
     +           frac(10),surf(10),points,rmvel,tmvel,u

c COMMON BLOCKS..........................................................

      COMMON/RECORDS  /cham,grve,aland,glr,twst,travp,igrad,chdist,
     +      chdiam,nchpts,prwt,htfr,pgas,iair,br,trav,npts,rcwt,rp,
     +      tr,nrp,ho,tshl,cshl,twal,hl,rhocs,forcig,covi,tempi,chwi,
     +      gamai,forcp,tempp,covp,chwp,rhop,gamap,nperfs,glenp,pdpi,
     +      pdpo,gdiap,dbpcp,nprop,alpha,pres,nbr,deltat,deltap,tstop,
     +      beta

      COMMON/S_RECORDS/s_cham,s_grve,s_aland,s_glr,s_twst,s_travp,
     +      s_igrad,s_chdist,s_chdiam,s_nchpts,s_prwt,s_htfr,s_pgas,
     +      s_iair,s_br,s_trav,s_npts,s_rcwt,s_rp,s_tr,s_nrp,s_ho,
     +      s_tshl,s_cshl,s_twal,s_hl,s_rhocs,s_forcig,s_covi,s_tempi,
     +      s_chwi,s_gamai,s_forcp,s_tempp,s_covp,s_chwp,s_rhop,
     +      s_gamap,s_nperfs,s_glenp,s_pdpi,s_pdpo,s_gdiap,s_dbpcp,
     +      s_nprop,s_alpha,s_tstop,s_beta,s_pres,s_nbr,s_deltat,
     +      s_deltap,s_bint,s_bvol,s_chmlen,s_bore

      COMMON/LIMITS   /dpmaxba,dpmaxbr,pmaxbr,pmaxba,max_mass,l_d,
     +      total_vol_prop,cham_vol

      COMMON/LOCALS   /bore,b1,b2,b3,b4,zz,bint,bvol,r1,r2,diam,areab,
     +      temp,chmlen,step,tmpi,lambda,pmaxm,tpmaxm,tpmaxbr,
     +      tpmaxba,tpmax,as,bs,ak,vp0,tr0,tcw,ibo,volgi,pmean,volg,
     +      wallt,ptime,z,y,points,ibrp,nde,isw1,area,resp,
     +      elpt,elpr,vzp,j4zp,elgpm,elbr,elrc,areaw,avcp,avc,avden,
     +      z18,z19,avvel,htns,elht,air,elar,rfor,eprop,rprop,tenergy,
     +      tgas,pt,v1,cov1,pbase,pbrch,j1zp,j2zp,j3zp,a2t,alf,a1t,bt,
     +      bata,gamma,delta,ds,p,t,rmvelo,tmvelo,disto,dfract,efi,efp,
     +      tenerg,tengas,frac,surf,rmvel,tmvel,i1,u

      COMMON/FILES    /outfil,bdfile

c end of intball.ins.f.....................................................
```

```
C***********************************************************************
      SUBROUTINE FUN_INT(X,NOVAR,FX,BURNED_UP)
C***********************************************************************
c     This subroutine is called from 'funx.ftn'. It is a modified
c  version of the lumped parameter interior ballistics code IBRGAC,
c  from the Interior Ballistics Laboratory, Maryland.  It has been
c  modified to accept iterative changes the input data. The following
c  changes have been made:
c
c    1. The input file is now read by an external subroutine,
c        'read_data.ftn'
c    2. The data is initialized by an external subroutine,
c        'reset_data.ftn'
c    3. Subroutine 'mass_check.ftn' checks the volume of propellent to
c         see if it will fit into the chamber.
c
c  See 'intball.ins.f' for variable definitions. In addition the following
c  defintions apply:
c    burned_up = the propellent is all burned up flag.
c    bad_web   = the flag for a web violation.
c **********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR,BURNED_UP
      REAL*4 X(NOVAR),FX,BAD_WEB

      BAD_WEB   = 0.0
      BURNED_UP = 0
      FX        = 0.0

C  RESET VARIABLES FOR RUN.............................................
      CALL reset_data

c  VARAIBLE ASSIGNMENT.................................................
      CALL VAR_IN(X,NOVAR,FX)
      IF (FX.LT.0.0) RETURN

c  CHECK MASS OF PROPELLENT............................................
      CALL MASS_CHECK

c  START INTERIOR BALLISTICS CALCULATIONS..............................
c  Calculate total mass of propellent and igniter.......................
      tmpi = 0.0
      do 20 i=1,nprop
20      tmpi = tmpi + chwp(i)
      tmpi = tmpi + chwi

c  Use Chambrage.......................................................
      if(igrad.gt.1) then
        go to 131
      else
c  Calculate the diameter of the bore [eq 1.3].........................
        bore = (glr*grve**2+aland**2)/(glr+1.0)
        bore = sqrt(bore)
      end if

c  Calculate the area of the bore......................................
131   areab   = pi*bore**2/4.0
c  Calculate the Nordheim Friction Factor [eq 7.15]....................
      lambda  = 1.0/((13.2+4.0*log10(100.0*bore))**2)

c  Initialization of Runge-Kutta values................................
      as(1) = 0.5
      as(2) = 1.-sqrt(2.)/2.
      as(3) = 1.+sqrt(2.)/2.
      as(4) = 1.0/6.0

      bs(1) = 2.0
      bs(2) = 1.0
      bs(3) = 1.0
      bs(4) = 2.0

      ak(1) = 0.5
      ak(2) = as(2)
      ak(3) = as(3)
      ak(4) = 0.5

      do 5 i = 1,nprop
        vp0    = chwp(i)/rhop(i)+vp0
5     continue
```

```
      volgi = cham - vp0 - chwi * covi
      pmean = forcig * chwi / volgi
      volg  = volgi
      volgi = volgi + vp0
      wallt = twal
      ptime = 0.0
      ibrp  = 8
      z(3)  = 1.0
      nde   = ibrp + nprop

      write(6,132)areab,pmean,vp0,volgi
132   format(1x,'area bore m^2 ',e16.6,' pressure from ign pa',e16.6,/

     +1x,' volume of unburnt prop m^3 ',e16.6,' init cham vol-cov ign m
     +^3 ',e16.6)
c       write(0,6)
c       write(6,6)
c6      format(1x,'   time        acc         vel         dis         mpress
c      +   pbase       pbrch     ')
        isw1=0
19    continue

      do 11 J=1,4
c.......................................................................
c  For loop the following applies to the z & y  arrays:
c  variable  defintion                 variable    defintion
c    z(1)      proj accel                y(1)       proj velocity
c    z(2)      proj velocity             y(2)       proj travel
c    z(3)      d(time)                   y(3)       time
c    z(4)      d(proj resistance         y(4)       proj resistance
c                energy)                              energy
c    z(5)      d(heat loss)              y(5)       heat loss
c    z(6)      recoil accel              y(6)       recoil velocity
c    z(7)      recoil velocity           y(7)       recoil travel
c    z(8)      d(air resistance          y(8)       energy loss from air
c                energy)                              resistance

c  FIND BARREL RESISTANCE..............................................
        do 201 k=2,npts
          if(y(2)+y(7).ge.trav(k)) then
            go to 201
          end if
          go to 203
201     continue

        k = npts

c  determine bore resistence due to friction and engraving [eq 7-8]
203     resp = (trav(k)-y(2)-y(7))/(trav(k)-trav(k-1))
        resp = br(k)-resp*(br(k)-br(k-1))

c  FIND MASS FRACTION BURNING RATE....................................
        do 211 k=1,nprop
          if(ibo(k).ne.1) then
c***********************************************************************
          CALL prf017(pdpo(k),pdpi(k),gdiap(k),dbpcp(k),glenp(k),
     +                surf(k),frac(k),y(ibrp+k),nperfs(k),u,bad_web)
c***********************************************************************
            if (bad_web.lt.0.0) then
              fx = -100.0*bad_web
              return
            end if
c  if surf is less than minimum then propellent all burned up..........
            if(surf(k).lt.1.e-10) ibo(k)=1
          end if
211     continue

c  ENERGY LOSS TO PROJECTILE TRANSLATION [eq 7-4].....................
        elpt=prwt*y(1)**2/2.0
```

```
c  ENERGY LOSS DUE TO PROJECTILE ROTATION [eq 7-5]....................
       elpr=pi**2*prwt*y(1)**2*twst**2/4.0

c  ENERGY LOSS DUE TO GAS AND PROPELLANT MOTION.......................
c  Chambrage
       if(igrad.eq.2) then
c net projectile travel
         pt  = y(2)+y(7)
c total current volume behind projectile
         vzp = bvol+areab*pt
c J4 determined at zp
         j4zp= bint(4)+((bvol+areab*pt)**3-bvol**3)/(3.0*areab**2)
c [eq 7.7]
         elgpm = tmpi*y(1)**2*areab**2*j4zp/(2.0*vzp**3)

c  Lagrange [eq 7.6]..................................................
       else
         elgpm = tmpi*y(1)**2/6.0
       end if

c  ENERGY LOSS FROM BORE RESISTANCE...................................
       elbr = y(4)
       z(4) = areab*resp*y(1)

c  ENERGY LOSS DUE TO RECOIL [eq 7.9].................................
       elrc = rcwt*y(6)**2/2.

c  ENERGY LOSS DUE TO HEAT LOSS.......................................
c [eq 7.13]
       areaw = cham/areab*pi*bore+2.0*areab+pi*bore*(y(2)+y(7))
       avden = 0.0
       avc   = 0.0
       avcp  = 0.0
       z18   = 0
       z19   = 0

       do 213 k=1,nprop
c  z18 is the left hand numerator term in eq 7.19
         z18   = forcp(k)*gamap(k)*chwp(k)*frac(k)/(gamap(k)-1.)
     +           /tempp(k)+z18
c  z19 is the left hand denominator term in eq 7.19
         z19   = chwp(k)*frac(k)+z19
c  the top left numerator term in eq 7.17
         avden = avden+chwp(k)*frac(k)
213    continue

c  [eq 7.19] specific heat at constant pressure of propellent gasses
       avcp  = (z18+forcig*gamai*chwi/(gamai-1.)/tempi)/(z19+chwi)
c  [eq 7.17] mean gas density
       avden = (avden+chwi)/(volg+cov1)
c  [eq 7.16] mean gas velocity
       avvel = .5*y(1)
c  [eq 7.14] Nordheim heat transfer coefficient
       htns  = lambda*avcp*avden*avvel+ho
c  [eq 7.12] Q dot
       z(5)  = areaw*htns*(tgas-wallt)*hl
c  [eq 7.11] heat loss
       elht  = y(5)
c  wall temperature
       wallt = (elht+htfr*elbr)/(cshl*rhocs*areaw*tshl)+twal

c  ENERGY LOSS DUE TO AIR RESISTANCE..................................
       air=iair
       z(8)=y(1)*pgas*air
       elar=areab*y(8)

c  RECOIL.............................................................
       z(6)=0.0
       if(pbrch.le.rp(1)/areab) then
         go to 221
       end if
       rfor=rp(2)
       if(y(3)-tr0.ge.tr(2)) then
         go to 222
       end if
       rfor = (tr(2)-(y(3)-tr0))/(tr(2)-tr(1))
       rfor = rp(2)-rfor*(rp(2)-rp(1))
222    z(6) = areab/rcwt*(pbrch-rfor/areab-resp)
       if(y(6).lt.0.0) then
         y(6) = 0.0
       else
```

```
            z(7) = y(6)
          end if
          goto 223
 221      tr0 = y(3)
 223      continue

 c  CALCULATE GAS TEMPERATURE............................................
          eprop = 0.0
          rprop = 0.0
          do 231 k=1,nprop
            eprop = eprop+forcp(k)*chwp(k)*frac(k)/(gamap(k)-1.)
 231        rprop = rprop+forcp(k)*chwp(k)*frac(k)/(gamap(k)-1.)/tempp(k)
          tenergy = elpt+elpr+elgpm+elbr+elrc+elht+elar
          tgas = (eprop + forcig*chwi/(gamai-1.0) - tenergy)/
      +          (rprop + forcig*chwi/((gamai-1.0)*tempi))

 c  FIND FREE VOLUME......................................................
          v1   = 0.0
          cov1 = 0.0
          do 241 k=1,nprop
            v1   = chwp(k)*(1.-frac(k))/rhop(k)+v1
            cov1 = cov1+chwp(k)*covp(k)*frac(k)
 241      continue
          volg = volgi+areab*(y(2)+y(7))-v1-cov1

 c     CALCULATE MEAN PRESSURE............................................
          r1 = 0.0
          do 251 k=1,nprop
 251        r1 = r1+forcp(k)*chwp(k)*frac(k)/tempp(k)

          pmean = tgas/volg*(r1+forcig*chwi/tempi)
          resp  = resp+pgas*air

          if(igrad.eq.2) then
            if(isw1.ne.0) then
              go to 253
            end if
            pbase = pmean
            pbrch = pmean
            if (pbase.gt.resp+1.0) then
              isw1 = 1
            end if
            go to 257

 c  USE CHAMBRAGE PRESSURE GRADIENT EQUATION..............................
 253        j1zp = bint(1)+(bvol*pt+areab/2.*pt**2)/areab
            j2zp = (bvol+areab*pt)**2/areab**2
            j3zp = bint(3)+areab*bint(1)*pt+bvol*pt**2/2.0+areab*pt**3/6.

            a2t   = -tmpi*areab**2/prwt/vzp**2
            alf   = 1.0-a2t*j1zp
            a1t   = tmpi*areab*(areab*y(1)**2/vzp+areab*resp/prwt)/vzp**2
            bt    = -tmpi*y(1)**2*areab**2/(2.0*vzp**3)
            bata  = -a1t*j1zp-bt*j2zp
            gamma = alf+a2t*j3zp/vzp
            delta = bata+a1t*j3zp/vzp+bt*j4zp/vzp

 c  CALCULATE BASE PRESSURE
            pbase = (pmean-delta)/gamma
 c  CALCULATE BREECH PRESSURE
            pbrch = alf*pbase+bata

          else
 c  USE LAGRANGE PRESSURE GRADIENT EQUATION...............................
 252        if(isw1.ne.0)go to 256

 c  CALCULATE BASE PRESSURE...............................................
 256        pbase=(pmean+tmpi*resp/3./prwt)/(1.+tmpi/3./prwt)
            if(pbase.gt.resp+1.) then
              isw1=1
            end if

 c  CALCULATE BREECH PESSURE..............................................
            pbrch = pbase+tmpi*(pbase-resp)/(2.0*prwt)
          end if
```

```
c  CALCULATE PROJECTILE ACCELERATICN...................................
         z(1) = areab*(pbase-resp)/prwt
         if(z(1).lt.0.0) then
           go to 257
         else
           go to 258
         end if
257      if(isw1.eq.0)   z(1) = 0.0
258      if(y(1).lt.0.0) y(1) = 0.0
         z(2)=y(1)

c  GET BURNING RATE....................................................
         do 264 m=1,nprop
           z(ibrp+m)=0.0
           if(ibo(m).eq.1) then
             goto 264
           end if
           do 262 k=1,nbr(m)
             if(pmean.gt.pres(m,k)) then
               go to 262
             end if
             go to 263
262        continue
           k=nbr(m)

c  [eq 5.2] linear burning rate........................................
263        z(ibrp+m)=beta(m,k)*(pmean*1.e-6)**alpha(m,k)
264      continue

c  4th order Runge-Kutta integration...................................
         do 21 i=1,nde
           ds(i) = (z(i)-bs(j)*p(i))*as(j)
           y(i) = deltat*ds(i)+y(i)
           p(i) = 3.*ds(i)-ak(j)*z(i)+p(i)
21       continue
11     continue

       t = t+deltat

c  set max mean pressure...............................................
       if(pmaxm.le.pmean) then
         pmaxm  = pmean
         tpmaxm = y(3)
       end if

c  set max base pressure...............................................
       if(pmaxba.le.pbase) then
         pmaxba  = pbase
         tpmaxba = y(3)
         dpmaxba = y(2)
       end if

c  set max breech pressure.............................................
       if(pmaxbr.le.pbrch) then
         pmaxbr  = pbrch
         tpmaxbr = y(3)
         dpmaxbr = y(2)
       end if

       if(y(3).ge.ptime) then
         ptime = ptime+deltap
c          write(0,7)y(3),z(1),y(1),y(2),pmean,pbase,pbrch
c          write(6,7)y(3),z(1),y(1),y(2),pmean,pbase,pbrch
c7          format(1x,7e11.4)
       end if

c  STOP CRITERIA: time is up or tube length is met.....................
       if(t.gt.tstop.or.y(2).ge.travp) then
         go to 200
       else
         rmvelo = y(1)
         disto  = y(2)
         tmvelo = y(3)
         goto 19
       end if
```

```
c  END OF CALCULATION OUTPUT.........................................
200   write(6,311)t,y(3)
311   format(1x,' deltat t', e14.6, ' intg t',e14.6)
      write(6,312)pmaxm,tpmaxm
312   format(1x,'PMAXMEAN Pa ',e14.6,' time at PMAXMEAN sec ',e14.6)
      write(6,313)pmaxba,tpmaxba
313   format(1x,'PMAXBASE Pa ',e14.6,' time at PMAXBASE sec ',e14.6)
      write(6,314)pmaxbr,tpmaxbr
314   format(1x,'PMAXBRCH Pa ',e14.6,' time at PMAXBRCH sec ',e14.6)
316   format(1x)

c  FX value assignment...............................................
c  it has either met the muzzle velocity criteria or current proj vel...
      if(y(2).le.travp) then
        write(6,327)y(1),y(3),y(2)-travp,l_d
327     format(1x,'proj VELOCITY m/s ',e14.6,' at time sec ',e14.6,
     +         '  tvl diff(-) =',f10.6,' d_l=',f10.4)
c  velocity check....................................................
        fx = -y(1)
        go to 319
      else
        dfract = (travp-disto)/(y(2)-disto)
        rmvel  = (y(1)-rmvelo)*dfract+rmvelo
        tmvel  = (y(3)-tmvelo)*dfract+tmvelo
c        write(6,318)rmvel,tmvel,y(2)-travp,l_d
c318     format(1x,'muzzle VELOCITY m/s ',e14.6,' at time sec ',e14.6,
c    +          '  tvl diff(+) =',f10.6,' p_f=',f10.4)
        fx = - rmvel
      end if

c  Energy calculations...............................................
319   efi = chwi*forcig/(gamai-1.)
      efp = 0.0
      do 315 i=1,nprop
        efp = efp + chwp(i) * forcp(i) / (gamap(i)-1.0)
315   continue

      tenerg = efi+efp
      write(6,317)tenerg
317   format(1x,'total initial energy available J = ',e14.6)
      tengas = chwi*forcig*tgas/(gamai-1.)/tempi
      do 135 i=1,nprop
        tengas=(frac(i)*chwp(i)*forcp(i)*tgas/tempp(i)/(gamap(i)-1.))
     +          + tengas
        write(6,328)i,frac(i)
328     format(' FOR PROPELLANT ',I2,' MASSFRACT BURNT IS ',e14.6)
135   continue

c  variable return...................................................
      CALL VAR_OUT(X,NOVAR)
      if (frac(1).ge.1.0) burned_up = 1

      return
      end

c  End of fun_int.ftn................................................
```

```
C**************************************************************************
      SUBROUTINE PRF017(P,P1,D,D1,L,SURF,MASSF,X,NP,U,BAD_WEB)
C**************************************************************************
c      This subroutine is called from 'fun_int.ftn'.  It is a modified
C  version of the 'PRF017' found in IBRGAC, from the Interior Ballistics
C  Laboratory, Maryland.  It has been modified only to ease understanding
C  of its operation. It performs the calculations of mass fraction and
C  surface fraction of propellent burned for zero, one, and seven
C  perforation propellent of unequal web.
C *************************************************************************
C  The following definitions apply.....................................
C    P        = OUTER PERF DIA
C    P1       = INNER PERF DIA
C    D        = OUTER DIA
C    D1       = DISTANCE BETWEEN PERF CENTERS
C    L        = GRAIN LENGTH
C    NP       = NUMBER OF PERFS
C    SURF     = OUTPUT SURFACE AREA
C    MASSF    = OUTPUT MASS FRACTION OF PROPELLANT BURNER
C    W        = WEB BETWEEN OUTER PERFS
C    WO       = OUTER WEB
C    W1       = WEB BETWEEN OUTER AND INNER PERFS
C    W4       = MINIMUM WEB
C    BAD_WEB = UNACCEPTABLE WE3B FLAG TO ENSURE FUNCTION VALUE
C               IS WEIGHTED
C    X        = DISTANCE THE PROPELLENT IS BURNED INTO SURFACE
C    U        = ORIGINAL VOLUME OF PROPELLENT GRAIN
C    A        = ORIGINAL SURFACE AREA OF 7 PERF PROPELLENT
C *************************************************************************

      INTEGER*2 NP
      REAL*4    pifor,d1sq,p12xsq,twopi,lm2x,p1sq,pp2x,pi,dm2xsq,d1sq3,
     +  d2sq3,pp2xsq,p1p2x,surf,a,d,twox,p,u,v,w,x,y,z,a1,a2,a3,a4,a5,
     +  b1,b2,b3,b4,b5,dsq,sqrt3,d1,f1,f2,f3,l1,l2,l3,hafpi,psq,p1,s0,
     +  s1,s2,v0,v1,v2,w0,xsq,w1,w4,x1,x2,massf,l,bad_web,dm2x

      DATA  PI,SQRT3/3.14159,1.732051/
      DATA  HAFPI,PIFOR,TWOPI/1.570796,.785398,6.283185/

      BAD_WEB = 0.0
      DSQ     = D*D
      PSQ     = P*P

      IF (NP.EQ.0) THEN
C ZERO PERF CALCULATIONS START HERE (CALC ORIGINAL VOLUME).............
        u = dsq*l*pi/4.
        IF (d-2*x.gt.0.0) THEN
          twox = x+x
          xsq  = x*x
          MASSF= TWOX*(DSQ+2.*L*D-4.*X*D-TWOX*L+4.*XSQ)/(DSQ*L)
          SURF = PI*(DSQ/2.-4.*D*X-TWOX*L+D*L+6.*XSQ)
C  PROPELLENT IS ALL BURNED UP.........................................
        ELSE
          MASSF = 1.0
          SURF  = 0.0
        END IF
        RETURN

      ELSE IF (NP.EQ.1) THEN
C ONE PERF CALCULATIONS START HERE (CALC ORIGINAL VOLUME)..............
        u = dsq*l*pi/4.-psq*l*pi/4.
        IF (d-p-4.*x.gt.0.0) THEN
          twox  = x+x
          MASSF = TWOX*(DSQ+2.*L*D-4.*X*D-PSQ+2.*P*L-4.*P*X)
     +            /(DSQ*L-PSQ*L)
          SURF  = PI*(DSQ/2.-4.*D*X-4.*X*P+D*L+P*L-PSQ/2.)
C  PROPELLENT IS ALL BURNED UP.........................................
        ELSE
          MASSF = 1.0
          SURF  = 0.0
        END IF
        RETURN
```

```
      ELSE IF (NP.EQ.7) THEN
C   SEVEN PERF PROPELLENT ACCEPTABLE DIMENSIONS CHECK....................
C   OUTER PERFORATION MIDPOINT BURNED THROUGH BY INNER PERF CHECK........
        IF(P1.GT.(P+D1*(SQRT3-1))) THEN
           bad_web = (P+D1*(SQRT3-1))-P1
C   OUTER PERFORATION MIDPOINT BURNED THROUGH BY OUTER DIAMETER CHECK....
        ELSE IF (D.LT.D1*(SQRT3+1.)-P) THEN
           BAD_WEB =D-(D1*(SQRT3+1.)-P)
        END IF

C   WEB DIMENSION CALCULATIONS..........................................
        W  = D1-P
        W0 = (D-P-2.*D1)/2.
        W1 = (2.*D1-P-P1)/2.
C   WEB BETWEEN OUTER PERF CHECK........................................
        IF (W.LT.0) THEN
          bad_web = w
C   OUTER WEB CHECK.....................................................
        ELSE IF (W0.LT.0.) THEN
          bad_web = w0

C   INNER WEB CHECK.....................................................
        ELSE IF(W1.LT.0.) THEN
          bad_web = w1
        END IF

C   UNACCPETABLE GRANULATION CHECK......................................
        IF (BAD_WEB.LT.0.0) GOTO 60

        P1SQ   = P1*P1
        D1SQ   = D1*D1
        D1SQ3 = D1*SQRT3
        D2SQ3 = D1SQ*SQRT3
        X1 = (P1SQ-PSQ+4.*D1SQ-2.*P1*D1SQ3)/4./(D1SQ3+P-P1)
        X2 = (4.*D1SQ+D*D-2.*D*D1SQ3-PSQ)/4./(-D1SQ3+P+D)
        A  = PI*L*(D+P1+6.*P)+HAFPI*(DSQ-P1SQ-6.*PSQ)
        U  = PI*L/4.*(DSQ-P1SQ-6.*PSQ)
        W4 = AMIN1(W,W0,W1)
        MASSF   = 0.0
        TWOX    = X+X
        XSQ     = X*X
        P1P2X   = P1+TWOX
        PP2X    = P+TWOX
        DM2X    = D-TWOX
        LM2X    = L-TWOX
        P12XSQ = P1P2X*P1P2X
        PP2XSQ = PP2X*PP2X
        DM2XSQ = DM2X*DM2X

C   SEVEN PERF CALCULATIONS START HERE..................................
C   IF LENGTH IS NOT ALL BURNED UP......................................
        IF (LM2X.GT.0) THEN
          S0 = PI*LM2X*(D+P1+6.*P+12.*X)+HAFPI*(DM2X*DM2X
     +         -P1P2X*P1P2X-6.*PP2X*PP2X)
          V0 = PIFOR*LM2X*(DM2X*DM2X-P1P2X*P1P2X-6.*PP2X*PP2X)
C   SEE IF SMALLEST WEB IS BURNED THROUGH...............................
          IF (X.GT.W4/2.) THEN
C   IF SO CHECK THE WEBS, ONE BY ONE....................................
C   FIRST CHECK INNER WEB...............................................
            IF (X.GT.W1/2.) THEN
C   IT IS BURNED UP.....................................................
              Z  = (2.*D1+P+P1+4.*X)/4.
              B3 = ((P1-P)*(P1+P+4.*X)+4.*D1SQ)/4./D1/P1P2X
              A3 = ATAN(SQRT(1.-B3*B3)/B3)
              B4 = ((P-P1)*(P+P1+4.*X)+4.*D1SQ)/4./D1/PP2X
              A4 = ATAN(SQRT(1.-B4*B4)/B4)
              F2 = A3/4.*P12XSQ+A4/4.*PP2XSQ-SQRT(Z*(Z-D1)*(2.*Z-P-TWOX)
     +             *(2.*Z-P1-TWOX))
              L2 = LM2X*(A4*PP2X+A3*P1P2X)
            ELSE
              F2 = 0.0
              L2 = 0.0
              A3 = 0.0
              A4 = 0.0
            END IF
```

```
C NEXT CHECK WEB BETWEEN OUTER PERFORATIONS.............................
            IF (X.GT.W/2.) THEN
              B5 = D1/PP2X
              A5 = ATAN(SQRT(1.-B5*B5)/B5)
              F3 = (A5*PP2XSQ-D1*SQRT(PP2XSQ-D1SQ))/2.
              L3 = 2.*A5*LM2X*PP2X
            ELSE
              F3 = 0.0
              L3 = 0.0
              A5 = 0.0
            END IF

C  NEXT CHECK OUTER WEB..................................................
            IF (X.GT.WO/2.) THEN
              Y  = (2.*D1+P+D)/4.
              B1 = ((D+P)*(D-P-4.*X)-4.*D1SQ)/4./D1/PP2X
              A1 = ATAN(SQRT(1.-B1*B1)/B1)
              IF (A1.LE.0.) A1 = PI+A1
              B2 = ((D+P)*(D-P-4.*X)+4.*D1SQ)/4./D1/DM2X
              A2 = ATAN(SQRT(1.-B2*B2)/B2)
              F1 = A1/4.*PP2XSQ-A2/4.*DM2XSQ+SQRT(Y*(Y-D1)*(2.*Y-P-TWOX)
     +             *(2.*Y-D+TWOX))
              L1 = LM2X*(A1*PP2X+A2*DM2X)
            ELSE
              F1 = 0.
              L1 = 0.
              A1 = 0.
              A2 = 0.
            END IF

C  ALL THREE WEBS HAVE BEEN CHECKED.....................................
C  DETERMINE SLIVERING EQUATIONS........................................

            IF (X.LE.W/2.) THEN
              SURF = S0+12.*(F1+F2+F3)-6.*(L1+L2+L3)
              V    = V0+6.*(F1+F2+F3)*LM2X
              GO TO 850
            END IF

            IF (X.LT.X1) THEN
              S1 = 3.*D2SQ3-PI*PP2XSQ-HAFPI*P12XSQ+6.*F3+12.*F2
              S1 = S1+LM2X*(2.*(PI-3.*A5-3.*A4)*PP2X+(PI-6.*A3)*P1P2X)
              V1 = LM2X/2.*(3.*D2SQ3-PI*PP2XSQ-HAFPI*P12XSQ+6.*F3
     +             +12.*F2)
              ELSE
              S1 = 0.0
              V1 = 0.0
            END IF

            IF (X.LT.X2) THEN
              S2 - HAFPI*DM2XSQ-3.*D2SQ3-TWOPI*PP2XSQ+12.*F1+6.*F3
              S2 = S2+LM2X*((PI-6.*A2)*DM2X+2.*(TWOPI-3.*A1-3.*A5)
     +             *PP2X)
              V2 = LM2X/2.*(HAFPI*DM2XSQ-3.*D2SQ3-TWOPI*PP2XSQ+12.
     +             *F1+6.*F3)
            ELSE
              S2 = 0.0
              V2 = 0.0
            END IF

            SURF = S1+S2
            V    = V1+V2

C  THE PROPELLENT HAS NO WEB VIOLATIONS................................
          ELSE
            MASSF = -TWOX/L/(DSQ-P1SQ-6.*PSQ)
            MASSF = MASSF*(24.*XSQ+(24.*P+4.*P1+4.*D-12.*L)*X+P1SQ
     +              +6.*PSQ-2.*L*D-2.*P1*L-12.*L*P-DSQ)
            SURF  = S0
            RETURN
          END IF
```

```
C  THE PROPELLENT HAS BEEN ALL BURNED UP................................
        ELSE
          SURF = 0.0
          V    = 0.0
        END IF

850     MASSF = 1.-V/U
        RETURN

C  THE NUMBER OF PERFORATIONS DOES NOT EQUAL 0,1 RO 7...................
      ELSE
   60   WRITE(6,90)
   90   FORMAT(1X,'UNACCEPTABLE GRANULATION')
      END IF

      RETURN
      END

c  end of prf017.........................................................
```

```
C*********************************************************************
      SUBROUTINE READ_DATA(X,NOVAR)
C*********************************************************************
c   This subroutine is called by 'optimum.ftn' and reads the input
c  records for 'fun_int.ftn'.  It saves all records into 's_' prefixed
C  variables.  At the end of this subroutine the inital design vector is
C  assigned.  See 'intball.ins.f' for variable definitions.
C*********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR

c  Input...............................................................
c  Input data file, and open file..
      write(*,15)
15    format('Enter name of data input file (10 characters max):')
      read(*,10)bdfile
10    format(a10)

      open(unit=2,err=30,file=bdfile,status='old',iostat=ios)
      rewind 2

c  Output data file, and open file.
      write(*,25)
25    format('Enter name of data output file (10 characters max):')
      read(*,10)outfil
      open(unit=6,err=30,file=outfil)
      write(6,16)bdfile

c  Read Record 1.......................................................
16    format('  The data input file is ',a10,/)
      read(2,*,end=20,err=30)s_cham,s_grve,s_aland,s_glr,s_twst,
     +  s_travp,s_igrad

c  Using chambrage gradient equation...................................
      if (s_igrad.gt.1) then
        write(6,47,err=30)
47      format(1x,' Using chambrage pressure gradient')

c  Read and echo print Record 1a (for chambrage only)..................
        read(2,*,end=20,err=30) s_nchpts,(s_chdist(I),s_chdiam(I),
     +    I=1,s_nchpts)
        write(6,53,err=30) (s_chdist(I),s_chdiam(I),I=1,s_nchpts)
53      format(///,'   chamber distance cm   chamber diameter cm',/
     +    (5x,e14.6,5x,e14.6))

c  convert units.......................................................
        do 54 I=1,s_nchpts
          s_chdist(I) = 0.01*s_chdist(I)
54        s_chdiam(I) = 0.01*s_chdiam(I)

c  calculate chamber integrals and volume..............................
        if (s_nchpts.gt.5) then
          s_nchpts = 5
          write(6,44,err=30)
44        format(1x,'use first 5 points')
        end if

c set bore to largest distance.........................................
        bore = s_chdiam(s_nchpts)

        if(s_chdist(1).ne.0.0) then
          write(6,45,err=30)
45        format(1x,' # points ? ')
        end if
        s_chdist(1) = 0.0

c  initialization......................................................
        b1 = 0.0
        b2 = 0.0
        b3 = 0.0
        b4 = 0.0

c  setting the initial number of integration points....................
        points = 25.0
56      points = points + points

c  setting the step size...............................................
        step = s_chdist(s_nchpts)/points
```

```
c  increment through breech/bore distance..............................
      zz = 0.0

c  initialize the bint's..............................................
      bint(1) = 0.0
      bint(3) = 0.0
      bint(4) = 0.0
      bvol    = 0.0

c  radius of the start of current interval.............................
      r2 = 0.5*s_chdiam(1)
      k = 1
      j = int(points+0.5)

c  going through the breech/bore travel................................
      do 57 I=1,j
        zz = zz+step

c if at last interval..................................................
        if (k.eq.s_nchpts-1) go to 46

c  looking for current interval in breech/bore.........................
        do 58 I1=k,s_nchpts-1
          if (zz.gt.s_chdist(I1).and. zz.lt.s_chdist(I1+1)) go to 59
58      continue
        I1 = s_nchpts-1
59      k  = I1

c  diam is first the ratio of the distance into the interval...........
46      diam = (zz-s_chdist(k))/(s_chdist(k+1)-s_chdist(k))

c  diam is then the diameter of the current location in the interval.
        diam = s_chdiam(k)+diam*(s_chdiam(k+1)-s_chdiam(k))

c  raduis at current location..........................................
        r1   = 0.5*diam

c  intermediate area of selected interval & step location..............
        area = pi*(r1+r2)**2/4.

c  the current net volume of the chamber...............................
        bvol = bvol+step*(pi/3.0)*(r1**2+r1*r2+r2**2)

c  calculating the current J(1), J(3), & J(4) values through numerical
c  integration.........................................................
        bint(1) = bint(1) + step * bvol / area
        bint(3) = bint(3) + step * area * bint(1)
        bint(4) = bint(4) + step * bvol**2 / area

c  set old radius to current radius....................................
57      r2 = r1

c  determing if convergence has been reached...........................
        temp   = abs(1.0-b1/bint(1))
        if(abs(1.0-b3/bint(3)).gt.temp) then
          temp = abs(1.0-b3/bint(3))
        end if
        if(abs(1.0-b4/bint(4)).gt.temp) then
          temp = abs(1.0-b4/bint(4))
        end if

c  if converged set values and exit....................................
        if(temp.le.0.001) then
          go to 41
        else
          b1 = bint(1)
          b3 = bint(3)
          b4 = bint(4)

c  or go back and close interval and try again.........................
          go to 56
        end if
```

```
c  convert units and save..................................................
41       s_cham    = bvol*1.e6
         s_chmlen  = s_chdist(s_nchpts)
         s_bint(1) = bint(1)
         s_bint(3) = bint(3)
         s_bint(4) = bint(4)
         s_bvol    = bvol
         s_bore    = bore

c  Use LaGrange Pressure Gradient.........................................
      else
        write(6,55)
55      format(1x,' Using Lagrange pressure gradient')
      end if

      write(6,40,err=30) s_cham,s_grve,s_aland,s_glr,s_twst,s_travp
40    format(1x,' chamber volume  cm^3',e14.6,/' groove diam    cm  ',
     +e14.6,/'  land diam       cm  ',e14.6,/' groove/land ratio   ',e1
     +4.6,/'  twist turns/caliber ',e14.6,/'  projectile travel cm',e14.
     +6/)

c  convert units..........................................................
      s_cham  = s_cham  * 1.0e-6
      s_grve  = s_grve  * 1.0e-2
      s_aland = s_aland * 1.0e-2
      s_travp = s_travp * 1.0e-2

c  Read and echo print Record 2...........................................
      read(2,*,end=20,err=30) s_prwt,s_iair,s_htfr,s_pgas
      write(6,50,err=30)      s_prwt,s_iair,s_htfr,s_pgas

50    format(1x,' projectile mass   kg',e14.6,/'  switch to calculate en
     +ergy lost to air resistance J',i3,/'  fraction of work against bor
     +e used to heat the tube',e14.6/1x,' gas pressure Pa',5x ,e14.6)

c  Read and echo print Record 3...........................................
      read(2,*,end=20,err=30) s_npts,(s_br(i),s_trav(i),i=1,s_npts)
       write(6,60,err=30)     s_npts,(s_br(i),s_trav(i),i=1,s_npts)
60     format(1x,' number barrel resistance points',i2,/'  bore resistan
     +ce MPa  -   travel cm'/(6x,e14.6,3x,e14.6))

      write(6,65)
65    format(1x)

c  convert units..........................................................
      do 62 i=1,s_npts
        s_br(i)   = s_br(i)   *1.0e6
        s_trav(i) = s_trav(i) *1.0e-2
62    continue

c  Read and echo print Record 4...........................................
      read(2,*,end=20,err=30) s_rcwt,s_nrp,(s_rp(i),s_tr(i),i=1,s_nrp)
       write(6,70,err=30)     s_rcwt,s_nrp,(s_rp(i),s_tr(i),i=1,s_nrp)
70     format(1x,' mass of recoiling parts   kg',5x,e14.6,/'  number of
     +recoil point pairs',6x,i2,/'  recoil force N','    recoil time sec
     +'/,(1x,e14.6,3x,e14.6))
      write(6,65)

c  Read and echo print Record 5...........................................
      read(2,*,end=20,err=30) s_ho,s_tshl,s_cshl,s_twal,s_hl,s_rhocs
       write(6,75,err=30)     s_ho,s_tshl,s_cshl,s_twal,s_hl,s_rhocs
75     format(1x,' free convective heat transfer coefficient w/cm^2 K  '
     +,e14.6,/'  chamber wall thickness cm',27x,e14.6,/'  heat capacity
     +of steel of chamber wall J/g K',8x,e14.6,/'  initial temperature o
     +f chamber wall K',15x,e14.6,/'  heat loss coefficient',31x,e14.6,/
     +'  density of chamber wall steel g/cm^3',16x,e14.6,//)

c  convert units..........................................................
      s_ho    = s_ho    /1.0e-4
      s_tshl  = s_tshl  *1.0e-2
      s_cshl  = s_cshl  *1.0e+3
      s_rhocs = s_rhocs *1.0e-3/1.0e-6

c  Read and echo print Record 6...........................................
      read(2,*,end=20,err=30) s_forcig,s_covi,s_tempi,s_chwi,s_gamai
       write(6,85,err=30)     s_forcig,s_covi,s_tempi,s_chwi,s_gamai
85     format(1x,' impetus of igniter propellant J/g',19x,e14.6,/'  covo
     +lume of igniter cm**3/g',25x,e14.6,/'  adiabatic flame temperature
     + of igniter propellant K ',e14.6,/'  initial mass of igniter kg',2
     +6x,e14.6,/'  ratio of specific heats for igniter',17x,e14.6//)
```

```
c  convert units.........................................................
      s_forcig = s_forcig*1.e+3
      s_covi   = s_covi*1.e-6/1.e-3

c  Read and echo print Record 7.........................................
      read(2,*,end=20,err=30)  s_nprop,(s_forcp(i),s_tempp(i),s_covp(i),
     +  s_chwp(i),s_rhop(i),s_gamap(i),s_nperfs(i),s_glenp(i),s_pdpi(i),
     +  s_pdpo(i),s_gdiap(i),s_dbpcp(i),i=1,s_nprop)
      write(6,95,err=30)   (i,s_forcp(i),s_tempp(i),s_covp(i),s_chwp(i),
     +  s_rhop(i),s_gamap(i),s_nperfs(i),s_glenp(i),s_pdpi(i),s_pdpo(i),
     +  s_gdiap(i),s_dbpcp(i),i=1,s_nprop)

95    format((' FOR PROPELLENT NUMBER',i2,/'  impetus of propellant J/g
     +',27x,e14.6,/'  adiabatic temperature of propellant K',16xe14.6,/'
     +  covolume of propellant cm**3/g',23x,e14.6/'  initial mass of pro
     +pellant kg',24x,e14.6/'  density of propellant g/cm**3',24x,e14.6/
     +'  ratio of specific heats for propellant',15x,e14.6/'  number of
     +perforations of propellant',18x,i2/'  length of propellant grain c
     +m',24x,e14.6/'  diameter of inner perforation in propellant grains
     + cm',e14.6/'  diameter of outer perforation of propellant grains c
     +m',e14.6/'  outside diameter of propellant grain cm',14x,e14.6/'
     +distance between perf centers cm',21x,e14.6)//)

c  convert units.........................................................
      do 96 i=1,s_nprop
        s_forcp(i) = s_forcp(i) *1.0e+3
        s_covp(i)  = s_covp(i)  *1.0e-6/1.0e-3
        s_rhop(i)  = s_rhop(i)  *1.0e-3/1.0e-6
        s_glenp(i) = s_glenp(i) *0.01
        s_pdpi(i)  = s_pdpi(i)  *0.01
        s_pdpo(i)  = s_pdpo(i)  *0.01
        s_gdiap(i) = s_gdiap(i) *0.01
        s_dbpcp(i) = s_dbpcp(i) *0.01
96    continue

c  Read and echo print Record 8.........................................
      do 97 j=1,s_nprop
        read(2,*,end=20,err=30) s_nbr(j),(s_alpha(j,i),s_beta(j,i),
     +    s_pres(j,i),i=1,s_nbr(j))
        write(6,110,err=30)     s_nbr(j),(s_alpha(j,i),s_beta(j,i),
     +    s_pres(j,i),i=1,s_nbr(j))
110     format(1x,' no. of burning rate points',i2/3x,'  exponent',5x,'
     +  coefficient',17x,'  pressure'/5x,'-',16x,'cm/sec-MPa**ai',12x,'
     +  MPa',/(1x,e14.6,5x,e14.6,15x,e14.6))

c  convert units.........................................................
        do 112 i=1,s_nbr(j)
          s_beta(j,i) = s_beta(j,i)*1.e-2
112       s_pres(j,i) = s_pres(j,i)*1.e6
97    continue

      write(6,65)

c  Read and echo print Record 9.........................................
      read(2,*,end=20,err=30) s_deltat,s_deltap,s_tstop
      write(6,120,err=30)     s_deltat,s_deltap,s_tstop
120   format(2x'time increment              msec',e14.6/2x,'print increment
     +            msec',e14.6/2x,'time to stop calculation msec',e14.6)

c  convert units.........................................................
      s_deltat = s_deltat *0.001
      s_deltap = s_deltap *0.001
      s_tstop  = s_tstop  *0.001

c  Design vector assigned here, each is problem specific................
c  Format is:     x(1) = s_glenp(1) etc.
c.......................................................................

129   write(0,130)
130   format(1x,' END INPUT DATA ')
      return

20    write(*,140)
140   format(1x,'end of file encounter')
      return

30    write(*,150)
150   format(1x,'read or write error')
      return
      end

c  end of data read.....................................................
```

```
c***********************************************************************
      SUBROUTINE RESET_DATA
c***********************************************************************
c    This subroutine resets the variables for and is called by
c  'fun_int.ftn'.  It transfers all initial values from their 's_'
c  prefixed variable to the variable that is used in the calling
c  subroutine. See 'intball.ins.f' for variable definitions.
c***********************************************************************
%INCLUDE 'intball.ins.f'

c  Record Initialization................................................
c  Record 1.............................................................
      cham  = s_cham
      grve  = s_grve
      aland = s_aland
      glr   = s_glr
      twst  = s_twst
      travp = s_travp
      igrad = s_igrad

c  Record 1a............................................................
      do 10 i=1,s_nchpts
        chdist(i) = s_chdist(i)
10      chdiam(i) = s_chdiam(i)
      nchpts = s_nchpts

c  Record 2.............................................................
      prwt = s_prwt
      htfr = s_htfr
      pgas = s_pgas
      iair  = s_iair

c  Record 3.............................................................
      do 30 i=1,s_npts
        br(i)   = s_br(i)
30      trav(i) = s_trav(i)
      npts = s_npts

c  Record 4.............................................................
      do 40 i=1,s_nrp
        rp(i) = s_rp(i)
40      tr(i) = s_tr(i)
      rcwt = s_rcwt
      nrp  = s_nrp

c  Record 5.............................................................
      ho    = s_ho
      tshl  = s_tshl
      cshl  = s_cshl
      twal  = s_twal
      hl    = s_hl
      rhocs = s_rhocs

c  Record 6.............................................................
      forcig = s_forcig
      covi   = s_covi
      tempi  = s_tempi
      chwi   = s_chwi
      gamai  = s_gamai

c  Record 7.............................................................
      do 70 i=1,s_nprop
        forcp(i) = s_forcp(i)
        tempp(i) = s_tempp(i)
        covp(i)  = s_covp(i)
        chwp(i)  = s_chwp(i)
        rhop(i)  = s_rhop(i)
        gamap(i) = s_gamap(i)
        nperfs(i)= s_nperfs(i)
        glenp(i) = s_glenp(i)
        pdpi(i)  = s_pdpi(i)
        pdpo(i)  = s_pdpo(i)
        gdiap(i) = s_gdiap(i)
70      dbpcp(i) = s_dbpcp(i)
      nprop = s_nprop
```

```
c  Record 8..........................................................
      do 80 i=1,10
        nbr(i) = s_nbr(i)
        do 80 j=1,10
          alpha(i,j) = s_alpha(i,j)
          beta(i,j)  = s_beta(i,j)
80        pres(i,j)  = s_pres(i,j)

c  Record 9..........................................................
      deltat = s_deltat
      deltap = s_deltap
      tstop  = s_tstop

c  LaGrange Chamber Volume resets....................................
      bint(1) = s_bint(1)
      bint(3) = s_bint(3)
      bint(4) = s_bint(4)

      bvol     = s_bvol
      chmlen   = s_chmlen
      bore     = s_bore

c  End of Record resets..............................................
c  Local use initialization..........................................
      r1       = 0.0
      r2       = 0.0
      areab    = 0.0
      tmpi     = 0.0
      lambda   = 0.0
      pmaxm    = 0.0
      pmaxbr   = 0.0
      pmaxba   = 0.0
      tpmaxm   = 0.0
      tpmaxbr  = 0.0
      tpmaxba  = 0.0
      tpmax    = 0.0
      air      = 0.0
      do 100 i=1,4
        as(i)  = 0.0
        bs(i)  = 0.0
100     ak(i)  = 0.0
      vp0   = 0.0
      tr0   = 0.0
      tcw   = 0.0
      volgi = 0.0
      pmean = 0.0
      volg  = 0.0
      wallt = 0.0
      ptime = 0.0
      do 110 i=1,20
        z(i) = 0.0
        y(i) = 0.0
        ds(i)= 0.0
110     p(i) = 0.0

      points = 0
      ibrp   = 0
      nde    = 0
      isw1   = 0
```

```
      resp    = 0.0
      elpt    = 0.0
      elpr    = 0.0
      pt      = 0.0
      vzp     = 0.0
      j4zp    = 0.0
      elgpm   = 0.0
      elbr    = 0.0
      elrc    = 0.0
      areaw   = 0.0
      avcp    = 0.0
      avc     = 0.0
      avden   = 0.0
      z18     = 0.0
      z19     = 0.0
      avvel   = 0.0
      htns    = 0.0
      elht    = 0.0
      elar    = 0.0
      rfor    = 0.0
      eprop   = 0.0
      rprop   = 0.0
      tenergy = 0.0
      tgas    = 0.0
      pt      = 0.0
      v1      = 0.0
      cov1    = 0.0
      pbase   = 0.0
      pbrch   = 0.0
      j1zp    = 0.0
      j2zp    = 0.0
      j3zp    = 0.0
      a2t     = 0.0
      alf     = 0.0
      a1t     = 0.0
      bt      = 0.0
      bata    = 0.0
      gamma   = 0.0
      delta   = 0.0
      t       = 0.0
      rmvelo  = 0.0
      tmvelo  = 0.0
      disto   = 0.0
      dfract  = 0.0
      efi     = 0.0
      efp     = 0.0
      tenerg  = 0.0
      tengas  = 0.0
      do 120 i=1,10
        frac(i) = 0.0
        surf(i) = 0.0
120     ibo(i)  = 0.0

      RETURN
      END

C  end of reset values..............................................
```

```
C*************************************************************************
      subroutine mass_check
C*************************************************************************
C  THIS SUBROUTINE DETERMINES THE LOADING DENSITY AND CHECKS THE MAXMIMUM
C  MASS ALLOWED FOR MULTIPLE PROPELLENTS
C*************************************************************************
%INCLUDE 'intball.ins.f'

      REAL*4 vol_grain(10),vol_ideal(10),mas_grain(10),num_grain(10),
     +       vol_prop(10),mas_prop(10),total_mas_prop

      intrinsic sqrt

c  check for propellent type............................................
      do 100 i=1,nprop
c  for 0 perf propellent................................................
        if (nperfs(i).eq.0) then
c  determine 1 grain volume, actual & ideal.............................
          vol_grain(i) = pi*glenp(i)*(gdiap(i)**2)/4.
          vol_ideal(i) = pi*glenp(i)*(gdiap(i)**2)/4.
c  determine the mass of 1 grain of propellent..........................
          mas_grain(i) = rhop(i)*vol_grain(i)
c  determine the number of grains of propellent present.................
          num_grain(i) = chwp(i)/mas_grain(i)
c  determine the pure volume of propellent present......................
          vol_prop(i)  = vol_ideal(i)*num_grain(i)
c  determine the mass of the propellent.................................
          mas_prop(i)  = mas_grain(i)*num_grain(i)

c  for 1 perf propellent................................................
        else if (nperfs(i).eq.1) then
          vol_grain(i) = pi*glenp(i)*(gdiap(i)**2-pdpi(i)**2)/4.
          vol_ideal(i) = pi*glenp(i)*(gdiap(i)**2)/4.
          mas_grain(i) = rhop(i)*vol_grain(i)
          num_grain(i) = chwp(i)/mas_grain(i)
          vol_prop(i)  = vol_ideal(i)*num_grain(i)
          mas_prop(i)  = mas_grain(i)*num_grain(i)

c  for 7 perf propellent................................................
        else if (nperfs(i).eq.7) then
          vol_grain(i) = pi*glenp(i)*(gdiap(i)**2-pdpi(i)**2-pdpo(i)**2)
     +                   /4.0
          vol_ideal(i) = pi*glenp(i)*(gdiap(i)**2)/4.
          mas_grain(i) = rhop(i)*vol_grain(i)
          num_grain(i) = chwp(i)/mas_grain(i)
          vol_prop(i)  = vol_ideal(i)*num_grain(i)
          mas_prop(i)  = mas_grain(i)*num_grain(i)
        end if
100   continue

c  the total volume the propellents occupy if ideally packed & mass....
      total_mas_prop = 0.0
      total_vol_prop = 0.0
      do 200 i=1,nprop
        total_mas_prop = total_mas_prob + chwp(i)
200     total_vol_prop = total_vol_prop + vol_prop(i)
c  the chamber volume (m^3) and loading density (g/cm^3)..............
      cham_vol = cham
      d_l = (total_mas_prop*1000.0)/(cham_vol*1e6)

      return
      end

c  end of check mass..................................................
```

```
c************************************************************************
      SUBROUTINE VAR_IN(x,novar,fx)
c************************************************************************
c   Example 1 variable conversion for Interior Ballistics Calculation
c************************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4 X(NOVAR),FX

c  Negative dimension check.  To ensure no negative values are sent...
      do 15 I=1,novar
15      if (x(i).le.0.0) fx = fx + x(i)
      if (fx.lt.0) then
        fx = -1000.0*fx
        return
      end if

c  variable assignment.............................................
c  7 perforation propellent
      glenp(1) = x(1)
      pdpi(1)  = x(2)
      pdpo(1)  = x(3)
      gdiap(1) = x(4)
      dbpcp(1) = x(5)
      chwp(1)  = x(6)

      return
      end
c  END OF VAR IN....................................................
```

```
c************************************************************************
      SUBROUTINE VAR_OUT(x,novar)
c************************************************************************
c   Example 1 variable return for Interior Ballistics Calculation
c************************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4    X(NOVAR)

c  variable return.................................................
c  perf = 7........................................................
      x(1) = glenp(1)
      x(2) = pdpi(1)
      x(3) = pdpo(1)
      x(4) = gdiap(1)
      x(5) = dbpcp(1)
      x(6) = chwp(1)

      RETURN
      END
c  END OF VAR OUT..................................................
```

```
c***********************************************************************
      SUBROUTINE VAR_IN(x,novar,fx)
c***********************************************************************
c   Example 2. This subroutine is called from 'fun_int.ftn' and sends
c  the design vector from the minimization process into the variables
c  used in the interior ballistics code.
c***********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4 X(NOVAR),FX

c  Negative dimension value check......................................
      do 15 I=1,novar
15      if (x(i).le.0.0) fx = fx + x(i)
      if (fx.lt.0) then
        fx = -1000.0*fx
        return
      end if


c  variable assignment................................................
c  7 perforation propellent
      glenp(1) = x(1)
      pdpi(1)  = x(2)
      pdpo(1)  = x(3)
      gdiap(1) = x(4)
      dbpcp(1) = x(5)
      chwp(1)  = x(9)
c  1 perforation propellent
      glenp(2) = x(6)
      pdpi(2)  = x(7)
      gdiap(2) = x(8)
      chwp(2)  = x(10)

      return
      end
c  END OF VAR IN.....................................................
```

```
c***********************************************************************
      SUBROUTINE VAR_OUT(x,novar,fx)
c***********************************************************************
c   Example 2. This subroutine is called from 'fun_int.ftn' and returns
c  the variables used in the interior ballistics code back into the
c  design vector format for the minimization process.
c***********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4    X(NOVAR)

c  variable return.....................................................
c  perf = 7
      x(1) = glenp(1)
      x(2) = pdpi(1)
      x(3) = pdpo(1)
      x(4) = gdiap(1)
      x(5) = dbpcp(1)
      x(9) = chwp(1)
c  perf = 1
      x(6) = glenp(2)
      x(7) = pdpi(2)
      x(8) = gdiap(2)
      x(10)= chwp(2)

      RETURN
      END
c  END OF VAR OUT.....................................................
```

```
c*********************************************************************
      SUBROUTINE VAR_IN(x,novar,fx)
c*********************************************************************
c   Example 3. This subroutine is called from 'fun_int.ftn' and sends
c  the design vector from the minimization process into the variables
c  used in the interior ballistics code.
c*********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4 X(NOVAR),FX

c  Negative dimension value check....................................
      do 15 I=1,novar
15      if (x(i).le.0.0) fx = fx + x(i)
      if (fx.lt.0) then
        fx = -1000.0*fx
        return
      end if

c  variable assignment...............................................
c  0 perforation propellent
      glenp(1) = x(1)
      gdiap(1) = x(2)
      chwp(1)  = x(11)
c  1 perforation propellent
      glenp(2) = x(3)
      pdpi(2)  = x(4)
      gdiap(2) = x(5)
      chwp(2)  = x(12)
c  7 perforation propellent
      glenp(3) = x(6)
      pdpi(3)  = x(7)
      pdpo(3)  = x(8)
      gdiap(3) = x(9)
      dbpcp(3) = x(10)
      chwp(3)  = x(13)

      return
      end

c  end of var in.....................................................

c*********************************************************************
      SUBROUTINE VAR_OUT(x,novar)
c*********************************************************************
c   Example 3. This subroutine is called from 'fun_int.ftn' and returns
c  the variables used in the interior ballistics code back into the
c  design vector format for the minimization process.
c*********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4    X(NOVAR)

c  design vector reassignment........................................
c  perf = 0..........................................................
      x(1) = glenp(1)
      x(2) = gdiap(1)
      x(11) = chwp(1)
c  perf = 1..........................................................
      x(3) = glenp(2)
      x(4) = pdpi(2)
      x(5) = gdiap(2)
      x(12) = chwp(2)
c  perf = 7..........................................................
      x(6) = glenp(3)
      x(7) = pdpi(3)
      x(8) = pdpo(3)
      x(9) = gdiap(3)
      x(10) = dbpcp(3)
      x(13) = chwp(3)

      RETURN
      END

c  End of var out....................................................
```

```
c*********************************************************************
      SUBROUTINE VAR_IN(x,novar,fx)
c*********************************************************************
c   Example 4. This subroutine is called from 'fun_int.ftn' and sends
c  the design vector from the minimization process into the variables
c  used in the interior ballistics code.
c*********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4 X(NOVAR),FX

c  Negative dimension value check.....................................
      do 15 I=1,novar
15      if (x(i).le.0.0) fx = fx + x(i)
      if (fx.lt.0) then
        fx = -1000.0*fx
        return
      end if


c  variable assignment................................................
c  7 perforation propellent
      glenp(1) = x(1)
      pdpi(1)  = x(2)
      pdpo(1)  = x(3)
      gdiap(1) = x(4)
      dbpcp(1) = x(5)
      chwp(1)  = x(11)
c  M8 7 perforation propellent
      glenp(2) = x(6)
      pdpi(2)  = x(7)
      pdpo(2)  = x(8)
      gdiap(2) = x(9)
      dbpcp(2) = x(10)
      chwp(2)  = x(12)

      return
      end
c  END OF VAR IN......................................................



c*********************************************************************
      SUBROUTINE VAR_OUT(x,novar,fx)
c*********************************************************************
c   Example 4. This subroutine is called from 'fun_int.ftn' and returns
c  the variables used in the interior ballistics code back into the
c  design vector format for the minimization process.
c*********************************************************************
%INCLUDE 'intball.ins.f'

      INTEGER*2 NOVAR
      REAL*4    X(NOVAR)

c  variable return....................................................
c  perf = 7
      x(1) = glenp(1)
      x(2) = pdpi(1)
      x(3) = pdpo(1)
      x(4) = gdiap(1)
      x(5) = dbpcp(1)
      x(11) = chwp(1)
c  M8 perf = 7
      x(6) = glenp(2)
      x(7) = pdpi(2)
      x(8) = pdpo(2)
      x(9) = gdiap(2)
      x(10) = dbpcp(2)
      x(12) = chwp(2)

      RETURN
      END
c  END OF VAR OUT.....................................................
```

APPENDIX III

INPUT FILES AND SAMPLE OUTPUT

This appendix contains a copy of the input file for each example and a example copy of the interior ballistic output file. A format guide is included after the input files to describe each entry. Each line in the input file corresponds to a record input.

Input file for problem 1a. 7-perforation propellent, sample

```
   9832.2384 12.7 12.7 1.0 0.0  457.2 1
   9.796 0 0.0 0.0
5  0.0 0.0 0.0 .6 0.0 1.3 0.0 300. 0. 457.
   1.e20 2 3.0e+4 0.0 8.0e+5 0.2
   .001135 .01143 .46028 273. 1. 7.8612
   84.5535 .9755 294. .004712 1.4
1  1135.99 3141. .9755 8.7 1.6605 1.23 7 3.175 .0508 .0508 1.0721 .2807
1  1.0 .1105187 689.476
   .005 .05 30.
```

Input file for problem 1b. 7-perforation propellent, sample

```
   9832.2384 12.7 12.7 1.0 0.0  457.2 1
   9.796 0 0.0 0.0
5  0.0 0.0 0.0 .6 0.0 1.3 0.0 300. 0. 457.
   1.e20 2 3.0e+4 0.0 8.0e+5 0.2
   .001135 .01143 .46028 273. 1. 7.8612
   84.5535 .9755 294. .004712 1.4
1  1135.99 3141. .9755 8.9 1.6605 1.23 7 4.00 .02 .04 2.0000 .400
1  1.0 .1105187 689.476
   .005 .05 30.
```

Input file for problem 2. 7-perforation propellent, sample
1-perforation propellent, sample

```
   9832.2384 12.7 12.7 1.0 0.0  457.2 1
   9.796 0 0.0 0.0
5  0.0 0.0 0.0 .6 0.0 1.3 0.0 300. 0. 457.
   1.e20 2 3.0e+4 0.0 8.0e+5 0.2
   .001135 .01143 .46028 273. 1. 7.8612
   84.5535 .9755 294. .004712 1.4
2  1135.99 3141. .9755 4.35 1.6605 1.23 7 3.175 .0508 .0508 1.0721 .2807
   1135.99 3141. .9755 4.35 1.6605 1.23 1 3.175 .0000 .0508 1.0721 .0000
1  1.0 .1105187 689.476
1  1.0 .1105187 689.476
   .005 .05 30.
```

Input file for problem 2. 0-perforation propellent, sample
1-perforation propellent, sample
7-perforation propellent, sample

```
   9832.2384 12.7 12.7 1.0 0.0  457.2 1
   9.796 0 0.0 0.0
5  0.0 0.0 0.0 .6 0.0 1.3 0.0 300. 0. 457.
   1.e20 2 3.0e+4 0.0 8.0e+5 0.2
   .001135 .01143 .46028 273. 1. 7.8612
   84.5535 .9755 294. .004712 1.4
3  1135.99 3141. .9755 3.00 1.6605 1.23 0 3.175 .0000 .0000 1.0721 .0000
   1135.99 3141. .9755 3.00 1.6605 1.23 1 3.175 .0000 .0508 1.0721 .0000
   1135.99 3141. .9755 3.00 1.6605 1.23 7 3.175 .0508 .0508 1.0721 .2807
1  1.0 .1105187 689.476
1  1.0 .1105187 689.476
1  1.0 .1105187 689.476
   .005 .05 30.
```

Input file for problem 4. 7-perforation propellent, sample
7-perforation propellent, M8

```
   9832.2384 12.7 12.7 1.0 0.0  457.2 1
   9.796 0 0.0 0.0
5  0.0 0.0 0.0 .6 0.0 1.3 0.0 300. 0. 457.
   1.e20 2 3.0e+4 0.0 8.0e+5 0.2
   .001135 .01143 .46028 273. 1. 7.8612
   84.5535 .9755 294. .004712 1.4
2  1135.99 3141. .9755 4.35 1.6605 1.23 7 3.175 .0508 .0508 1.0721 .2807
   1168.90 3768. .9550 4.35 1.2119 1.62 7 3.175 .0508 .0508 1.0721 .2807
1  1.0 .1105187 689.476
1  1.0 .1105187 689.476
   .005 .05 30.
```

```
c     Format for input files.  Each line in file is 1 record.  Record 1a is read
c     only if IGRAD = 2 in Record 1.
c     Record 1.............................................................
c         CHAM        REAL*4       CHAMBER VOLUME                          cm^3
c         GRVE        REAL*4       GROOVE DIAMETER                         cm
c         ALAND       REAL*4       LAND DIAMETER                           cm
c         GLR         REAL*4       GROOVE/LAND RATIO                       none
c         TWST        REAL*4       TWIST                               turns/caliber
c         TRAVP       REAL*4       PROJECTILE TRAVEL                       cm
c         IGRAD       INTEGER*2    GRADIENT FLAG                           none
c                                  1 = Lagrange,  2 = Chambrage
c     Record 1a............................................................
c         NCHPTS      INTEGER*2    NUMBER POINTS TO DESCRIBE CHAMBER       none
c       For I=1,nchp ts
c         CHDIST(I)   REAL*4       INITIAL DISTANCE FROM BREECH            cm
c         CHDIAM(I)   REAL*4       DIAMETER AT CHDIST(I)                   cm
c     Record 2.............................................................
c         PWRT        REAL*4       PROJECTILE MASS                         kg
c         IAIR        INTEGER*2    CALCULATE ENERGY LOST TO AIR
c                                    RESISTANCE FLAG  (1 = yes)            none
c         HTFR        REAL*4       FRACTION OF WORK DONE AGAINST
c                                    BORE TO HEAT TUBE                     none
c         PGAS        REAL*4       GAS PRESSURE IN FRONT OF PROJECTILE     Pa
c     Record 3.............................................................
c         NPTS        INTEGER*2    NUMBER OF BARREL RESISTANCE POINTS      none
c       For I=1,npts
c         BR(I)       REAL*4       BORE RESISTANCE                         MPa
c         TRAV(I)     REAL*4       TRAVEL                                  cm
c     Record 4.............................................................
c         RCWT        REAL*4       MASS OF RECIOLING PARTS                 kg
c         NRP         INTEGER*2    NUMBER OF RECOIL PAIR POINTS            none
c       For I=1,nrp
c         RP(I)       REAL*4       RECOIL FORCE                            N
c         TR(I)       REAL*4       RECOIL TIME                             s
c     Record 5.............................................................
c         HO          REAL*4       FREE CONVECTION HEAT TRANSFER
c                                    COEFFICIENT                           w/cm^2-k
c         TSHL        REAL*4       CHAMBER WALL THICKNESS                  cm
c         CSHL        REAL*4       HEAT CAPACITY OF STEEL OF CHAMBER WALL  J/g-k
c         TWAL        REAL*4       INITIAL TEMPERATURE OF CHAMBER WALL     k
c         HL          REAL*4       HEAT LOSS COEFFICIENT                   none
c         RCHOS       REAL*4       DENSITY OF CHAMBER WALL STEEL           g/cm^3
c     Record 6.............................................................
c         FORCIG      REAL*4       IMPETUS OF IGNITER PROPELLENT           J/g
c         COVI        REAL*4       COVOLUME OF IGNITER                     cm^3/g
c         TEMPI       REAL*4       ADIABATIC FLAME TEMP OF IGNITER         k
c         CHWI        REAL*4       INITIAL MASS OF IGNITER                 kg
c         GAMAI       REAL*4       RATIO OF SPECIFIC HEAT FOR IGNITER      none
c     Record 7.............................................................
c         NPROP       INTEGER*2    NUMBER OF PROPELLENT TYPES              none
c       For I=1,nprop
c         FORCP(I)    REAL*4       IMPETUS OF PROPELLENT                   J/g
c         TEMPP(I)    REAL*4       ADIABATIC TEMPERATURE OF PROPELLENT     k
c         COVP(I)     REAL*4       COVOLUME OF PROPELLENT                  cm^3/g
c         CHWP(I)     REAL*4       INITIAL MASS OF PROPELLENT              kg
c         RHOP(I)     REAL*4       DENSITY OF PROPELLENT                   g/cm^3
c         GAMAP(I)    REAL*4       RATIO OF SPECIFIC HEATS, PROPELLENT     none
c         NPERFS(I)   INTEGER*2    NUMBER OF PERFORATIONS ON PROPELLENT    none
c         GLENP(I)    REAL*4       LENGTH OF PROPELLENT GRAIN              cm
c         PDPI(I)     REAL*4       DIAMETER OF INNER PERFORATIONS
c                                    IN PROPELLENT GRAIN                   cm
c         PDPO(I)     REAL*4       DIAMETER OF OUTER PERFORATIONS
c                                    IN PROPELLENT GRAIN                   cm
c         ODIAP(I)    REAL*4       OUTSIDE DIAMETER OF PROPELLENT GRAIN    cm
c         DBPCP(I)    REAL*4       DISTANCE BETWEEN PERFORATION CENTERS    cm
c     Record 8.............................................................
c       For J=1,nprop
c         NBR(J)      INTEGER*2  NUMBER OF BURNING POINTS                  none
c       For I=1,nbr(j)
c         ALPHA(J,I)  REAL*4       EXPONENT                                none
c         BETA(J,I)   REAL*4       COEFFICIENT                             cm/s-MPa
c         PRESS(J,I)  REAL*4       PRESSURE                                MPa
c     Record 9.............................................................
c         DELTAT      REAL*4       TIME INCREMENT (STEP)                   ms
c         DELTAP      REAL*4       PRINT INCREMENT                         ms
c         TSTOP       REAL*4       STOP TIME FOR CALCULATIONS              ms
```

```
The data input file is p3final.in

Using Lagrange pressure gradient
chamber volume  cm^3  0.983224E+04
groove diam     cm    0.127000E+02
land diam       cm    0.127000E+02
groove/land ratio     0.100000E+01
twist turns/caliber   0.000000E+00
projectile travel cm  0.457200E+03

projectile mass   kg  0.979600E+01
switch to calculate energy lost to air resistance J  0
fraction of work against bore used to heat the tube  0.000000E+00
gas pressure Pa       0.000000E+00
number barrel resistance points 5
bore resistance MPa  -  travel cm
      0.000000E+00     0.000000E+00
      0.000000E+00     0.600000E+00
      0.000000E+00     0.130000E+01
      0.000000E+00     0.300000E+03
      0.000000E+00     0.457000E+03

mass of recoiling parts   kg        0.100000E+21
number of recoil point pairs        2
recoil force N    recoil time sec
 0.300000E+05     0.000000E+00
 0.800000E+06     0.200000E+00

free convective heat transfer coefficient w/cm^2 K     0.113500E-02
chamber wall thickness cm                              0.114300E-01
heat capacity of steel of chamber wall J/g K           0.460280E+00
initial temperature of chamber wall K                  0.273000E+03
heat loss coefficient                                  0.100000E+01
density of chamber wall steel g/cm^3                   0.786120E+01


impetus of igniter propellant J/g                      0.845535E+02
covolume of igniter cm**3/g                            0.975500E+00
adiabatic flame temperature of igniter propellant K    0.294000E+03
initial mass of igniter kg                             0.471200E-02
ratio of specific heats for igniter                    0.140000E+01


FOR PROPELLENT NUMBER 1
impetus of propellant J/g                               0.113599E+04
adiabatic temperature of propellant K                   0.314100E+04
covolume of propellant cm**3/g                          0.975500E+00
initial mass of propellant kg                           0.992000E+00
density of propellant g/cm**3                           0.166050E+01
ratio of specific heats for propellant                  0.123000E+01
number of perforations of propellant                    0
length of propellant grain cm                           0.249350E+01
diameter of inner perforation in propellant grains cm   0.000000E+00
diameter of outer perforation of propellant grains cm   0.000000E+00
outside diameter of propellant grain cm                 0.128500E+01
distance between perf centers cm                        0.000000E+00


FOR PROPELLENT NUMBER 2
impetus of propellant J/g                               0.113599E+04
adiabatic temperature of propellant K                   0.314100E+04
covolume of propellant cm**3/g                          0.975500E+00
initial mass of propellant kg                           0.362600E+01
density of propellant g/cm**3                           0.166050E+01
ratio of specific heats for propellant                  0.123000E+01
number of perforations of propellant                    1
length of propellant grain cm                           0.594250E+01
diameter of inner perforation in propellant grains cm   0.000000E+00
diameter of outer perforation of propellant grains cm   0.000000E+00
outside diameter of propellant grain cm                 0.579800E+00
distance between perf centers cm                        0.000000E+00
```

FOR PROPELLENT NUMBER 3

impetus of propellant J/g	0.113599E+04
adiabatic temperature of propellant K	0.314100E+04
covolume of propellant cm**3/g	0.975500E+00
initial mass of propellant kg	0.413000E+01
density of propellant g/cm**3	0.166050E+01
ratio of specific heats for propellant	0.123000E+01
number of perforations of propellant	7
length of propellant grain cm	0.599600E+01
diameter of inner perforation in propellant grains cm	0.180000E-02
diameter of outer perforation of propellant grains cm	0.180000E-02
outside diameter of propellant grain cm	0.861300E+00
distance between perf centers cm	0.217000E+00

no. of burning rate points 1

exponent	coefficient	pressure
-	cm/sec-MPa**ai	MPa
0.100000E+01	0.110519E+00	0.689476E+03

no. of burning rate points 1

exponent	coefficient	pressure
-	cm/sec-MPa**ai	MPa
0.100000E+01	0.110519E+00	0.689476E+03

no. of burning rate points 1

exponent	coefficient	pressure
-	cm/sec-MPa**ai	MPa
0.100000E+01	0.110519E+00	0.689476E+03

time increment msec 0.500000E-02
print increment msec 0.500000E-01
time to stop calculation msec 0.300000E+02
area bore m^2 0.126677E-01 pressure from ign pa 0.873844E+05
volume of unburnt prop m^3 0.526829E-02 init cham vol-cov ign m ^3 0.982764E-02

time	acc	vel	dis	mpress	pbase	pbrch
0.1050E-03	0.1111E+03	0.1036E-01	0.5216E-06	0.1115E+06	0.8590E+05	0.1243E+06
0.2500E-03	0.1540E+03	0.2943E-01	0.3331E-05	0.1546E+06	0.1191E+06	0.1723E+06
0.4050E-03	0.2154E+03	0.5783E-01	0.9970E-05	0.2162E+06	0.1666E+06	0.2410E+06
0.5550E-03	0.2933E+03	0.9573E-01	0.2134E-04	0.2943E+06	0.2268E+06	0.3281E+06
0.7000E-03	0.3891E+03	0.1449E+00	0.3862E-04	0.3905E+06	0.3009E+06	0.4353E+06
0.8500E-03	0.5129E+03	0.2122E+00	0.6518E-04	0.5148E+06	0.3966E+06	0.5738E+06
0.1000E-02	0.6661E+03	0.3003E+00	0.1033E-03	0.6685E+06	0.5151E+06	0.7452E+06
0.1150E-02	0.8538E+03	0.4138E+00	0.1565E-03	0.8569E+06	0.6603E+06	0.9552E+06
0.1300E-02	0.1083E+04	0.5585E+00	0.2290E-03	0.1087E+07	0.8375E+06	0.1212E+07
0.1450E-02	0.1363E+04	0.7412E+00	0.3260E-03	0.1368E+07	0.1054E+07	0.1525E+07
0.1600E-02	0.1704E+04	0.9704E+00	0.4537E-03	0.1710E+07	0.1318E+07	0.1907E+07
0.1750E-02	0.2122E+04	0.1256E+01	0.6199E-03	0.2130E+07	0.1641E+07	0.2374E+07
0.1900E-02	0.2633E+04	0.1612E+01	0.8341E-03	0.2643E+07	0.2037E+07	0.2946E+07
0.2050E-02	0.3261E+04	0.2052E+01	0.1108E-02	0.3273E+07	0.2522E+07	0.3648E+07
0.2200E-02	0.4031E+04	0.2597E+01	0.1455E-02	0.4046E+07	0.3117E+07	0.4510E+07
0.2355E-02	0.5012E+04	0.3295E+01	0.1910E-02	0.5030E+07	0.3876E+07	0.5607E+07
0.2505E-02	0.6181E+04	0.4132E+01	0.2464E-02	0.6203E+07	0.4779E+07	0.6915E+07
0.2655E-02	0.7614E+04	0.5163E+01	0.3159E-02	0.7642E+07	0.5888E+07	0.8519E+07
0.2805E-02	0.9371E+04	0.6432E+01	0.4025E-02	0.9405E+07	0.7247E+07	0.1048E+08
0.2955E-02	0.1152E+05	0.7994E+01	0.5103E-02	0.1156E+08	0.8909E+07	0.1289E+08
0.3105E-02	0.1415E+05	0.9912E+01	0.6441E-02	0.1420E+08	0.1094E+08	0.1583E+08
0.3255E-02	0.1735E+05	0.1227E+02	0.8098E-02	0.1741E+08	0.1342E+08	0.1941E+08
0.3405E-02	0.2124E+05	0.1515E+02	0.1015E-01	0.2131E+08	0.1642E+08	0.2376E+08
0.3555E-02	0.2594E+05	0.1868E+02	0.1268E-01	0.2604E+08	0.2006E+08	0.2903E+08
0.3705E-02	0.3162E+05	0.2298E+02	0.1579E-01	0.3173E+08	0.2445E+08	0.3537E+08
0.3855E-02	0.3842E+05	0.2822E+02	0.1962E-01	0.3856E+08	0.2971E+08	0.4298E+08
0.4005E-02	0.4651E+05	0.3457E+02	0.2431E-01	0.4668E+08	0.3597E+08	0.5204E+08
0.4155E-02	0.5608E+05	0.4225E+02	0.3006E-01	0.5628E+08	0.4336E+08	0.6274E+08
0.4305E-02	0.6727E+05	0.5148E+02	0.3706E-01	0.6751E+08	0.5202E+08	0.7526E+08
0.4455E-02	0.8021E+05	0.6251E+02	0.4559E-01	0.8050E+08	0.6202E+08	0.8973E+08
0.4605E-02	0.9497E+05	0.7563E+02	0.5592E-01	0.9532E+08	0.7344E+08	0.1063E+09
0.4755E-02	0.1116E+06	0.9110E+02	0.6840E-01	0.1120E+09	0.8627E+08	0.1248E+09
0.4905E-02	0.1298E+06	0.1092E+03	0.8338E-01	0.1303E+09	0.1004E+09	0.1453E+09
0.5050E-02	0.1489E+06	0.1294E+03	0.1006E+00	0.1495E+09	0.1152E+09	0.1666E+09
0.5200E-02	0.1697E+06	0.1533E+03	0.1218E+00	0.1703E+09	0.1312E+09	0.1898E+09
0.5350E-02	0.1910E+06	0.1803E+03	0.1468E+00	0.1917E+09	0.1477E+09	0.2137E+09
0.5500E-02	0.2122E+06	0.2106E+03	0.1761E+00	0.2130E+09	0.1641E+09	0.2374E+09
0.5650E-02	0.2326E+06	0.2439E+03	0.2101E+00	0.2334E+09	0.1799E+09	0.2602E+09
0.5800E-02	0.2515E+06	0.2803E+03	0.2494E+00	0.2524E+09	0.1945E+09	0.2814E+09
0.5950E-02	0.2682E+06	0.3193E+03	0.2943E+00	0.2692E+09	0.2074E+09	0.3001E+09

0.6100E-02	0.2823E+06	0.3606E+03	0.3453E+00	0.2834E+09	0.2183E+09	0.3159E+09
0.6250E-02	0.2935E+06	0.4038E+03	0.4026E+00	0.2946E+09	0.2270E+09	0.3284E+09
0.6400E-02	0.3016E+06	0.4485E+03	0.4665E+00	0.3027E+09	0.2332E+09	0.3374E+09
0.6550E-02	0.3067E+06	0.4942E+03	0.5372E+00	0.3078E+09	0.2372E+09	0.3432E+09
0.6700E-02	0.3090E+06	0.5404E+03	0.6148E+00	0.3102E+09	0.2390E+09	0.3457E+09
0.6850E-02	0.3089E+06	0.5867E+03	0.6993E+00	0.3100E+09	0.2388E+09	0.3455E+09
0.7000E-02	0.3066E+06	0.6329E+03	0.7908E+00	0.3077E+09	0.2371E+09	0.3430E+09
0.7150E-02	0.3025E+06	0.6786E+03	0.8892E+00	0.3036E+09	0.2339E+09	0.3384E+09
0.7300E-02	0.2970E+06	0.7236E+03	0.9943E+00	0.2981E+09	0.2297E+09	0.3323E+09
0.7450E-02	0.2905E+06	0.7677E+03	0.1106E+01	0.2916E+09	0.2247E+09	0.3250E+09
0.7600E-02	0.2832E+06	0.8107E+03	0.1225E+01	0.2843E+09	0.2190E+09	0.3169E+09
0.7750E-02	0.2754E+06	0.8526E+03	0.1349E+01	0.2764E+09	0.2130E+09	0.3081E+09
0.7900E-02	0.2673E+06	0.8933E+03	0.1480E+01	0.2683E+09	0.2067E+09	0.2990E+09
0.8050E-02	0.2590E+06	0.9328E+03	0.1617E+01	0.2600E+09	0.2003E+09	0.2898E+09
0.8200E-02	0.2505E+06	0.9710E+03	0.1760E+01	0.2514E+09	0.1937E+09	0.2802E+09
0.8350E-02	0.2395E+06	0.1008E+04	0.1909E+01	0.2403E+09	0.1852E+09	0.2679E+09
0.8500E-02	0.2274E+06	0.1043E+04	0.2062E+01	0.2282E+09	0.1759E+09	0.2544E+09
0.8650E-02	0.2152E+06	0.1076E+04	0.2221E+01	0.2160E+09	0.1664E+09	0.2408E+09
0.8800E-02	0.2033E+06	0.1107E+04	0.2385E+01	0.2040E+09	0.1572E+09	0.2274E+09
0.8950E-02	0.1918E+06	0.1137E+04	0.2553E+01	0.1925E+09	0.1483E+09	0.2146E+09
0.9105E-02	0.1807E+06	0.1166E+04	0.2732E+01	0.1813E+09	0.1397E+09	0.2021E+09
0.9255E-02	0.1706E+06	0.1192E+04	0.2909E+01	0.1712E+09	0.1319E+09	0.1909E+09
0.9405E-02	0.1612E+06	0.1217E+04	0.3090E+01	0.1618E+09	0.1247E+09	0.1804E+09
0.9555E-02	0.1525E+06	0.1241E+04	0.3274E+01	0.1531E+09	0.1179E+09	0.1706E+09
0.9705E-02	0.1444E+06	0.1263E+04	0.3462E+01	0.1450E+09	0.1117E+09	0.1616E+09
0.9855E-02	0.1369E+06	0.1284E+04	0.3653E+01	0.1374E+09	0.1059E+09	0.1532E+09
0.1000E-01	0.1299E+06	0.1304E+04	0.3847E+01	0.1304E+09	0.1005E+09	0.1454E+09
0.1015E-01	0.1225E+06	0.1323E+04	0.4044E+01	0.1230E+09	0.9475E+08	0.1371E+09
0.1030E-01	0.1156E+06	0.1341E+04	0.4244E+01	0.1160E+09	0.8939E+08	0.1293E+09
0.1045E-01	0.1093E+06	0.1358E+04	0.4446E+01	0.1097E+09	0.8449E+08	0.1222E+09

```
deltat time   0.105500E-01            intg time     0.105498E-01
PMAXMEAN Pa   0.310367E+09 time at PMAXMEAN sec     0.676509E-02
PMAXBASE Pa   0.239143E+09 time at PMAXBASE sec     0.676509E-02
PMAXBRCH Pa   0.345980E+09 time at PMAXBRCH sec     0.676509E-02
Muzzle VELOCITY (m/s)  0.136756E+04 at time sec     0.105473E-01
```

```
Total Initial Energy Available J =   0.432081E+08
FOR PROPELLANT 1 MASSFRACT BURNT =   0.487515E+00
FOR PROPELLANT 2 MASSFRACT BURNT =   0.100000E+01
FOR PROPELLANT 3 MASSFRACT BURNT =   0.100000E+01
```