Missile Design Toolbox

Ekker, David A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MISSILE DESIGN TOOLBOX

by

David A. Ekker

December 1994

Thesis Advisor: Conrad F. Newberry
Second Reader: Richard M. Howard

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1.AGENCY USE ONLY *(Leave blank)*	2.REPORT DATE	3.REPORT TYPE AND DATES COVERED
	December 1994	Engineer's Thesis

4.TITLE AND SUBTITLE

MISSILE DESIGN TOOLBOX

5.FUNDING NUMBERS

6.AUTHOR(S)

Ekker, David A.

7.PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey CA 93943-5000

8.PERFORMING ORGANIZATION REPORT NUMBER

9.SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11.SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a.DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13.ABSTRACT ***(maximum 200 words)***

Missile Design (MSLDSN) is a toolbox created to be used with The MathWorks' MATLAB® interactive computing environment, version 4.2. MSLDSN provides the missile designer with a number of tools to aid in establishing and evaluating missile parameters during the conceptual phase of design. MSLDSN aids the designer in establishing an initial configuration which is then refined using the various missile programs (e.g., Missile Datcom). MSLDSN is limited to designing solid-propellant-powered, tail-controlled air-to-air missiles with triangular planform fins in a cruciform arrangement.

14. SUBJECT TERMS	15.NUMBER OF PAGES
Aerodynamics, Missile Design	253
	16.PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19.SECURITY CLASSIFICATION OF ABSTRACT	20.LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited

MISSILE DESIGN TOOLBOX

David A. Ekker
Lieutenant Commander, United States Navy
B.S.M.E., University of Illinois, 1983

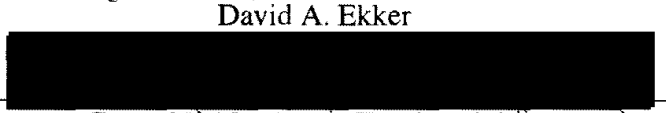
Submitted in partial fulfillment of the
requirements for the degree of

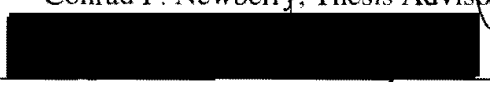
AERONAUTICAL AND ASTRONAUTICAL ENGINEER

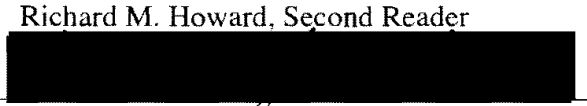
from the

NAVAL POSTGRADUATE SCHOOL
December 1994

Author: 
David A. Ekker

Approved by: 
Conrad F. Newberry, Thesis Advisor


Richard M. Howard, Second Reader


Daniel J. Collins, Chairman

Department of Aeronautics and Astronautics

ABSTRACT

Missile Design (MSLDSN) is a toolbox created to be used with The MathWorks' MATLAB® interactive computing environment, version 4.2. MSLDSN provides the missile designer with a number of tools to aid in establishing and evaluating missile parameters during the conceptual phase of design. MSLDSN aids the designer in establishing an initial configuration which is then refined using the various missile programs (e.g., Missile Datcom). MSLDSN is limited to designing solid-propellant-powered, tail-controlled air-to-air missiles with triangular planform fins in a cruciform arrangement.

TABLE OF CONTENTS

NOMENCLATURE

SYMBOL	DESCRIPTION	UNITS
a	acceleration	ft/sec^2
a	acceleration	ft/sec^2
A	aspect ratio	-
A_p	planform area	ft^2
AR	aspect ratio	-
b	exposed span	ft
c_{dc}	cross-flow drag coefficient of the cylindrical section	-
c_r	root chord	ft
CG	center of gravity	-
$(C_{DB})_\alpha$	body drag coefficient due to angle of attack	-
C_L	lift coefficient	-
C_m	pitching moment coefficient	-
d	body diameter	ft
D	drag force	lb
$\overline{D}$	average drag force	lb
D	distance	ft
deg	degrees	deg
DENSITY	total missile density	lb/ft^3
DIAMETER	missile and subsection diameter	ft
E	elliptical integral of the second kind	-
F_z	external force along the Z axis	lb
ft	feet	ft
g	gravitational constant = 32.174	ft/sec^2
GCLEN	guidance/control subsection length	ft

GCVOL	guidance/control subsection volume	ft^3
GCWT	guidance/control subsection weight	lb
h	height	ft
$H(t)$	unit step function	-
I_y	moment of inertia about the Y axis	slugs-ft^2
I_{sp}	specific impulse	sec
k	ratio of lift component to lift of wing alone for variable wing incidence	-
$(k_2 - k_1)$	Munk's fineness ratio factor. k_2 is Munk's longitudinal apparent mass coefficient. k_1 is Munk's transverse apparent mass coefficient.	-
K	ratio of lift component to lift of wing alone for variable angle of attack	-
K	constant	-
l	length of missile, length of a component	ft
l	distance measured from the tip of the body's nose	ft
l_T	distance from the tip of the nose to the intersection of tail leading edge and body	ft
l_W	distance from the tip of the nose to the intersection of wing leading edge and body	ft
ℓ	over-all length from wing apex to most aft point on trailing edge	ft
lb	pound	lb
L	lift force	lb
LENGTH	total missile length	ft
m	mass	slugs
	cotangent of leading edge sweep angle	-

M	Mach number	-
	moment	ft-lb
M_y	external moment about the Y axis	ft-lb
MAC	mean aerodynamic chord	ft
n	load factor	-
N	normal force	lb
NM	nautical mile	NM
P	pressure coefficient	-
PLEN	propulsion subsection weight	ft
PVOL	propulsion subsection volume	ft^3
PWT	propulsion subsection weight	lb
$\overline{q}$	dynamic pressure	slug/ft-sec
q	pitch rate about the Y axis	radians/sec
$\dot{q}$	pitching acceleration about the Y axis	radians/sec^2
r	body radius	ft
r	sample correlation coefficient	-
r^2	coefficient of determination	-
R	radius	ft
RANGE	range	NM
Re	Reynolds number	-
s	maximum semispan of wing in combination with body	ft
sec	seconds	sec
S	cross-sectional area, surface area	ft^2
SWP	leading edge sweep angle	degrees
t	thickness	ft
t	time	sec
$t \rightarrow \infty$	as time approaches infinity	sec
Δt	time increment	sec

T	thrust	lb
TR	taper ratio	-
V	volume	ft^3
VOL	volume	ft^3
VOLUME	total missile volume	ft^3
V	velocity	ft/sec
ΔV_b	incremental velocity due to boost	ft/sec
W	weight	lb
WEIGHT	total missile weight	lb
WFWT	wing/fin subsection weight	lb
WHLEN	warhead subsection length	ft
WHVOL	warhead subsection volume	ft^3
WHWT	warhead subsection weight	lb
WT	weight	lb
x	distance measured from the center of gravity	ft
x	characteristic length	ft
$\left(\frac{x}{c_r}\right)$	center of pressure location in fraction of root chord	-
$\overline{X}$	static stability margin	ft
α	angle of attack	radian or deg
β	$\sqrt{\lvert M^2-1\rvert}$	-
δ	control surface deflection	radian or deg
ε	downwash angle	radian or deg
γ	flight path angle	radians
$\dot{\gamma}$	flight path angular rate	radians/sec
Λ	sweep angle	radian or deg

η	efficiency	-
	drag proportionality factor	-
κ	lift efficiency	-
λ	taper ratio	-
μ	coefficient of viscosity	slug/ft-sec
v_p	volumetric packing factor	-
θ	elevation angle about the Y axis	radians
$\dot{\theta}$	elevation angle rate about the Y axis	radians/sec
$\ddot{\theta}$	elevation angle acceleration about the Y axis	radians/sec^2
ρ	density	slug/ ft^3
σ	semi-vertex angle	deg
ω	frequency	radians/sec
ζ	damping factor	-
ζ_m	rocket motor mass ratio	-
∞	infinity	

SUBSCRIPTS

0	zero lift
b	base
b	boost
B	body
B(W)	body in the presence of the wing
B(T)	body in the presence of the tail
cg	center of gravity
comp	component
cp	center of pressure
c/2	half chord

c/4	quarter chord
C	cruise
d	damped
D	drag
D	divergence
eb	end-burning
E	empty
f	final
f	friction
H	horizontal
i	induced
i	incompressible
ib	internal burning
L	launch
LE	leading edge
m	missile
m	motor
max	maximum
n	increment
nz	nozzle
N	nose
o	initial
p	peak
P	pressure
P	propellant
ref	reference
s	sustain

T	tail
T	target
T(B)	tail in the presence of the body
TR	trim condition
V	vertical
W	wing
WB	wing-body combination
W(B)	wing in the presence of the body
WBT	wing-body-tail combination
WET	wetted
W	wave
α	signifies partial derivative with respect to angle of attack, parameter value as a result of angle of attack
δ	signifies partial derivative with respect to angle of incidence, parameter value as a result of incidence angle

I. INTRODUCTION

Missile Design Toolbox (MSLDSN) is a series of files written to be used with The MathWorks' MATLAB® interactive computing environment, version 4.2. MSLDSN provides the missile designer with a number of tools to aid the user in establishing and evaluating missile configuration parameters during the conceptual phase of design. A basic understanding of the MATLAB language is assumed.

A. BACKGROUND AND PURPOSE

There exist a number of computer programs, such as Advanced Design of Aerodynamic Missiles (Hindes, 1993) and Missile Datcom (Bruns et al, 1991) which can assist a designer in analyzing a missile design. These programs are extremely powerful and important tools. Unfortunately, most of the programs are highly specialized and thus cover only one area of the design process, e.g., aerodynamics or trajectory analysis. Additionally, the programs require significant decision making by the designer before they can be executed. Once program execution is complete, the output is presented in its own unique format. If the data are to be used by another program, the user may be required to either laboriously reenter the data by hand or to reformat it. These programs may lose much of their effectiveness in the early phases of the design process when such reformatting takes an inordinate amount of the designer's time and effort.

MSLDSN attempts to aid the designer in establishing an initial configuration which can be refined during the design process by using the various MSLDSN sub-programs. For example, in Chapter II of this work, an initial weight estimate for the missile is made from an initial estimate of diameter, length and range. Only those three inputs are necessary to give an approximate estimate for the missile subsection weights and lengths, which can be used to calculate an initial center of gravity; one of the inputs required for calculating aerodynamic derivatives. The center of gravity location is used along with an estimate of missile maneuverability requirements to calculate wing size,

wing location and tail size in Chapter III. By the end of the Chapter III discussion, the user of MSLDSN should have the initial missile dimensions defined.

The first priority in the development of the MSLDSN methodology was to select a suitable programming environment. The environment needs to handle math- intensive operations, provide graphics capabilities and be able to evaluate control system designs. The last requirement narrowed the field markedly.

After evaluating a number of computing environments, the author selected The MathWorks' MATLAB for MSLDSN. MATLAB provides a user-friendly and flexible environment in which to conduct computations. In addition to the capabilities of the basic program, toolboxes are available from The MathWorks and other sources which enhance MATLAB's abilities. A toolbox is a series of files written in MATLAB's m-file format to assist the user to solve problems in a given subject area. The Statistics, Symbolic, Control System, Optimization and SIMULINK toolboxes are examples of toolboxes available. These additional toolboxes are not required to run the majority of MSLDSN. Where necessary, reference is made to the specific toolboxes required.

Before starting the actual programming in MATLAB, the author determined the scope and philosophy of MSLDSN. With the tendency of modern missiles to operate at high angles of attack (Fleeman, 1992), the tail-controlled missile is a good candidate for an air-to-air missile design. Missiles with canard control reach control surface saturation more rapidly than tail control and thus their angle of attack is limited. The tail-controlled missile has additional advantages of smaller control actuator requirements and more linear aerodynamic characteristics than a missile with wing control. Currently both the Sidewinder and Sparrow replacements under development or proposed use a tail-controlled configuration. Thus, in scope, MSLDSN is limited to designing solid-propellant-powered, tail-controlled air-to-air missiles with triangular planform fins in a cruciform arrangement. The cruciform arrangement was selected over a monowing due to the monowing's requirement to bank to turn and thus not maneuver as rapidly. The triwing configuration was not selected since a larger wing size is required to develop the

same aerodynamic forces as the cruciform configuration. The larger wing size negates the drag gain of fewer surfaces (Lindsey et al, 1980). Figure 1 displays the assumed generic arrangement of the missile. The basic philosophy of the program is to require the user to input as few variables as possible, but allow the user easy access to the output data for analysis. The user retains the capability to modify any assumptions made in the development of the program.

B. INSTALLING THE TOOLBOX

MSLDSN was developed using MATLAB version 4.2c for Microsoft Windows®. To use MSLDSN, the m-files (filenames with *.m extension) that make up MSLDSN need to be copied to their own directory and the MATLAB path m-file, MATLABRC.M,

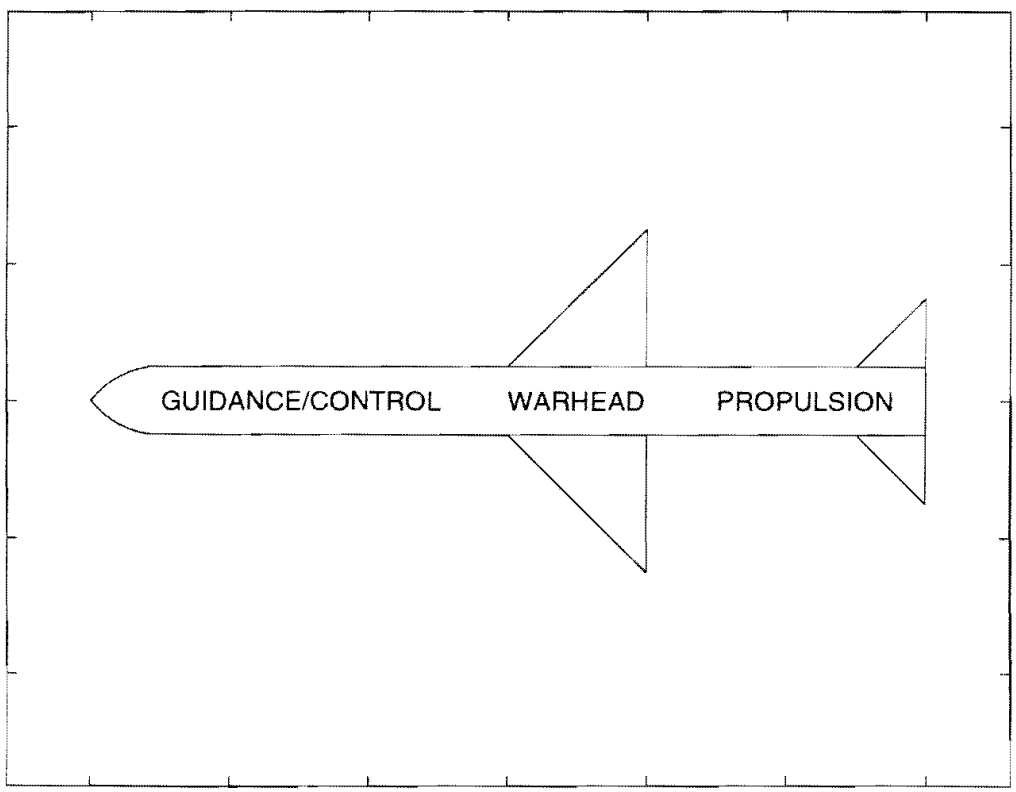


Figure 1. Assumed Missile Configuration

needs to be edited by the user to include the new directory. The MATLAB User's Guide (Little and Moler, 1993) provides information on MATLAB path and MATLABRC.M.

For example, assuming the user has installed MATLAB in the default directory 'C:\MATLAB', the user should create the directory:

'C:\MATLAB\TOOLBOX\MSLDSN'

using the file manager and copy all the files from the MSLDSN diskette to this directory. The user should then start MATLAB and open the m-file MATLABRC.M using the 'Open M-file' option in the 'File' Command Window menu. Following the line 'matlabpath([...' will be a number of lines of the form:

';C:\MATLAB\toolbox\matlab\datafun',...

These lines establish the path where the MATLAB program searches for files when MATLAB is running. The user appends the line:

';C:\MATLAB\toolbox\msldsn',...

before the path closing line, ']);'. The user should then close the file to save the changes and restart MATLAB for the change to take effect.

M-file file names are capitalized in the text to help distinguish them from variables and are presented without the *.m extension. To get help with respect to an individual m-file, the user should type 'help' followed by the name of the m-file, in lower-case letters, at the MATLAB prompt. For example, to get help on MATLABRC, the user should type 'help matlabrc' at the prompt.

```
» help matlabrc
```

The following help information is displayed on the screen:

```
MATLABRC Master startup M-file.

    MATLABRC is automatically executed by MATLAB during startup.
```

It establishes the MATLAB path, sets the default figure size, and sets a few uicontrol defaults.

On multi-user or networked systems, the system manager can put any messages, definitions, etc. that apply to all users here.

MATLABRC also invokes a STARTUP command if the file 'startup.m' exists on the MATLAB path.

A summary of m-files in MSLDSN is available to the user and accessed by typing 'help' followed by the name of the subdirectory where the MSLDSN m-files are stored. For example, if the m-files are located in the directory 'C:\MATLAB\TOOLBOX\MSLDSN', by typing 'help msldsn' at the prompt, a summary of m-files is returned to the screen and the user can peruse them.

C. HOW TO USE MSLDSN

Each chapter of this thesis starts with an introduction followed by a detailed explanation of the methodologies used. Examples are provided to help familiarize the user with MSLDSN use. The best way to learn to use MSLDSN is to review the introductory material of each chapter, skim the details and then run the sample problems. Instead of entering commands every time, the user should become familiar with the writing of script and function files. An excellent tutorial on the subject is provided in the MATLAB User's Guide (Little and Moler, 1993) starting on page 2-128.

The detail of the text is presented to allow the user to verify whether the approach used is sufficiently valid for the application. A familiarity with terms used in missile design and aircraft design texts is assumed.

The symbols used in the text follow the nomenclature found in the reference documents as closely as possible. The variables used in MATLAB cannot always exactly follow the nomenclature of the text due to the inability of MATLAB to assign subscripts

and Greek symbols as variables. Most variables are easily recognizable when taken in context. For example, the wing lift-curve-slope is represented in the text as $C_{L\alpha W}$, whereas in the m-files the variable for wing lift-curve-slope is CLaW. The definition of each m-file's input and output variables can be found using the 'help' command.

II. MISSILE WEIGHT ESTIMATION

In the early stages of design, estimates need to be made of the missile component weights and dimensions. Regression formulas developed from historical data are available to make a first approximation (Nowell, 1992). The assumption is made when applying the regression technique that the validity of the missile configuration parameters is justified during the missile design process. By applying these regression formulae, the weight and approximate length of a missile can be readily developed. Later in the design process, more detailed information is gained for each component and the missile dimensions can be refined until the final missile dimensions are established.

At the end of this Chapter, the weight and length of each of the major subsections of the generic missile will be determined along with an estimation of the location of the center of gravity. The required inputs from the user are the missile diameter, approximate missile length and range.

Since the user may want to review data used by Nowell in developing his formulae, the missile data are available to the user in the m-files AAMDUS, AAMDUSGC, AAMDUSP, AAMDUSWF, AAMDUSWH and AAMDWW. The user should try 'help aamdus' to get a summary of the US air-to-air missile data.

A. SUBSECTION DESCRIPTION

Nowell (1992) divided the missile into three body subsections and a wing/fin subsection. Depending on the missile, the components of each subsection may include:

1. Propulsion

- Power plant
- Peripherals required to support the plant
- Nozzle
- Case body
- Fuel
- Fuel tanks

- Auxiliary power units
- Air intakes

2. **Guidance and Control**
   - Radome
   - Control actuators for wings and fins
   - Seeker
   - Autopilot
   - Gyroscope
   - Data processor
   - Antenna
   - Inertial measurement unit
   - Radar receiver/transmitter
   - Power supply (battery)
   - Case body
3. **Warhead**
   - Payload (explosive and casing)
   - Fuze or target detection device
   - Safety device
   - Arming device
   - Case body
4. **Wing/Fin**
   - Exposed portion of the wing or fin

B. STATISTICAL REGRESSION

Mathematical regression analysis requires establishing a database and then attempting to relate one dependent variable to one or more independent variables. To help explain the development process, a simple linear regression example is developed herein, using the regression techniques in Beyer (1981).

If the following linear relationship exists between the independent variable x and a dependent variable y. Where b1 and b0 are, as of yet, undetermined coefficients and e is a normally distributed random variable.

$$y = b0 + b1x + e \tag{2.1}$$

The least squares estimates of b1 and b0 are:

$$b1 = \frac{n\sum x_i y_i - \left(\sum x_i\right)\left(\sum y_i\right)}{n\sum x_i^2 - \left(\sum x_i\right)^2} \tag{2.2}$$

$$b0 = \frac{\sum y_i - b1\sum x_i}{n} \tag{2.3}$$

To evaluate the degree to which the data fit the simple linear regression model, the coefficient of determination, r^2, is calculated. The square root of r^2, r, is the sample correlation coefficient. The sample correlation coefficient is a measure of how strongly related x and y are in the sample. The coefficient of determination has a value from 0 to 1. The higher the value of r^2, or closer to 1, the better the linear model fits the data.

$$r^2 = 1 - \frac{\sum y_i^2 - b0\sum y_i - b1\sum x_i y_i}{\sum y_i^2 - (\sum y_i)^2 / n} \tag{2.4}$$

By using mathematical transformations, the simple linear regression model can be used to investigate other relationships. The power law relationship:

$$y = a\,x^b \tag{2.5}$$

can be transformed into a linear model by using logarithms:

$$\log y = \log a + b \log x \tag{2.6}$$

The m-file RGRSDEMO investigates the relationship between air-to-air missile volume (VOL) to weight (WT) using simple linear regression models. To run RGRSDEMO, the user types:

```
» rgrsdemo
```

Which returns:

RGRSDEMO investigates the relationship between air-to-air missile volume (VOL) to weight (WT) using simple linear regression models. Two relationships will be examined:

$$WT = b1*VOL + b0$$

and

$$WT = a*VOL^{b}$$

The m-files RGRSSLIN and RGRSSPOW will be applied to the data from the m-file AAMDWW.

Coefficients

b1	b0	r^2
57.4262	135.9637	0.8508

a	b	
141.9252	0.7371	0.9328

Thus, the power law model correlates these particular data better than does the linear model.

To see how RGRSDEMO works, the user should either open the RGRSDEMO m-file and review the code or review the copy of the m-file in Appendix B. Table 1 summarizes the results from RGRSDEMO. In this particular case, the higher value of the coefficient of determination for the power law model implies that the power law model fits the data better than does the linear model. Thus the preferred relationship between volume to weight would be:

$$WT = 141.9252\ VOL^{0.7371} \tag{2.7}$$

Numerous and more complex relationships can be employed by using multiple independent variables. In addition to the coefficient of determination, the standard error of estimation and the plot of residuals should be examined to determine valid

b1	b0	r^2
57.4262	135.9637	0.8508
a	b	
141.9252	0.7371	0.9328

Table 1. Comparison of Linear Versus Power Regression Model

relationships. A more detailed explanation of regression can be found in most references on statistics, e.g. Hogg (1977) or Devore (1987).

C. AIR-TO-AIR MISSILE WEIGHT

1. Methodology

The m-file AAMBSLN uses regression formulas from Nowell (1992) to estimate the subsection weights and lengths for an air-to-air missile with an initial data input of missile range, length and diameter. The methodology for subsection weight estimates starts with calculating an initial missile weight and density from the input information. Using this initial information, subsection weight, volume and length are calculated. Calculating the weight for the Wing/Fin is done slightly differently and the details are addressed in Subsection e of this Section. The final missile weight and length are composed of the individual subsection weights and lengths.

a. Initial Volume, Weight and Density

$$\text{VOLUME} = \frac{\pi}{4}\text{DIAMETER}^2\,\text{LENGTH} \tag{2.8}$$

$$\text{WEIGHT} = 142.2\,\text{VOLUME}^{0.74} \tag{2.9}$$

$$\text{DENSITY} = \frac{\text{WEIGHT}}{\text{VOLUME}} \tag{2.10}$$

b. Propulsion Subsection

$$\text{PWT} = -284.9 + 633.6\,\text{DIAMETER} - 0.105\,\text{WEIGHT} + 0.949\,\text{DENSITY} \tag{2.11}$$

$$\text{PVOL} = \frac{\text{PWT} - 2.7}{112} \tag{2.12}$$

$$\text{PLEN} = \frac{4}{\pi}\frac{\text{PVOL}}{\text{DIAMETER}^2} \tag{2.13}$$

c. Guidance/Control Subsection

$$GCWT = 117.6\ DIAMETER + 1.6\ RANGE - 0.14\ DENSITY \tag{2.14}$$

$$GCVOL = \left(\frac{GCWT}{83.9}\right)^{1.5873} \tag{2.15}$$

$$GCLEN = \frac{4}{\pi}\frac{GCVOL}{DIAMETER^2} \tag{2.16}$$

d. Warhead Subsection

$$WHWT = 0.1\ DENSITY - 0.2\ RANGE + 0.2\ WEIGHT - 2.4\ LENGTH \tag{2.17}$$

$$WHVOL = \left(\frac{WHWT}{103.9}\right)^{1.2821} \tag{2.18}$$

$$WHLEN = \frac{4}{\pi}\frac{WHVOL}{DIAMETER^2} \tag{2.19}$$

e. Wing/Fin Weight

The weight of one fin is first calculated and then multiplied by eight to obtain the total weight for all the fins, eight being the number of fins for all of the air-to-air missiles in the database. Thus, if the user desires to try a triform configuration, the total weight of the fins would still be calculated by multiplying the weight of one fin by eight. Since the weight of the fins is highly dependent on RANGE, SWP, AR and TR, representative or average values from the missile database, AAMDUSWF, were used as initial estimates.

For short range missiles, RANGE < 20 NM, the following average values and equations were used:

SWP = 45 (DEG)

AR = 2.07

TR = 0.63

$$WFWT = 5.4 + 0.005\ WEIGHT - 0.2\ SWP + 11.1\ TR \tag{2.20}$$

For medium range missiles, RANGE 20 - 50 NM:

SWP = 55 (DEG)

AR = 2.38

$$WFWT = -89.8 + 0.03\ WEIGHT + 0.99\ SWP + 13\ AR \quad (2.21)$$

For long range missiles, RANGE > 50 NM:

$$SWP = \quad 84\ (DEG)$$

$$AR = \quad 0.56$$

$$WFWT = 1.3\ AR + 0.1\ SWP + 0.0006\ WEIGHT \quad (2.22)$$

The database used to estimate the subsection weights is composed of only five US air-to-air missiles. The amount of unclassified information on missile component data is limited. Similarly, data on missiles of foreign manufacture is limited. The further the initial missile inputs diverge from the size parameters of one of these missiles, the less accurate will be the subsection weight and length estimates. AAMBSLN also outputs the initial missile weight using a regression formula based on all air-to-air missiles in the database. This weight is based on a database of twenty air-to-air missiles from all over the world. If this weight significantly varies from the sum of the subsection weights, the user should review the individual subsection weights to see if they seem reasonable.

2. AAMBSLN Examples

As a familiarization exercise, the user should try AAMBSLN with AMRAAM's dimensions and range as the input for designing a medium range air-to-air missile (MRAAM). To find out exactly what inputs are required, the user should start with help AAMBSLN:

```
» help aambsln
```

Which returns:

```
AAMBSLN Calculates the baseline parameters for an air-to-air missile using
regression formulas.
        baseline = AAMBSLN(diameter,length,range,dsplyoff)
```

diameter = missile diameter (FT)

length = missile length (FT)

range = missile range(NM)

dsplyoff = enter any numerical value to turn off the display summary sent to the screen

baseline = [weight length wgtfinal lenfinal gcwt gclen whwt whlen pwt plen wfwt]

weight = initial weight estimate

length = initial length estimate

wgtfinal = total weight based on sum of subsections

lenfinal = total length based on sum of subsections

gcwt = guidance/control subsection weight

gclen = guidance/control subsection length

whwt = warhead subsection weight

whlen = warhead subsection length

pwt = propulsion subsection weight

plen = propulsion subsection length

wfwt = total wing/fin weight

Thus, the required input is estimated missile diameter, length and range. From the m-file AAMDUS, summary data on the AMRAAM is obtained. The user should type 'help aamdus' to review the data.

```
»help aamdus
```

AAMDUS lists the following data for the AMRAAM:

- DIAMETER 0.6 FT

- LENGTH 12 FT
- RANGE 35 NM

The user runs AAMBSLN by entering the following at the MATLAB prompt:

```
» bsln=aambsln(0.6,12,35)
```

Returning:

```
Initial Estimate
 Weight   Length
   lb       ft
351.1754  12.0000

        Total          Guidance/Control       Warhead
 Weight   Length   Weight    Length   Weight   Length
   lb       ft       lb        ft       lb       ft
362.4674  11.6624  112.0697  5.5998  44.7853  1.2023

     Propulsion     Wing/Fin
 Weight   Length   Weight
   lb       ft       lb
156.6104  4.8602  49.0021

bsln =
 Columns 1 through 7
 351.1754  12.0000  362.4674  11.6624  112.0697   5.5998  44.7853
 Columns 8 through 11
  1.2023  156.6104   4.8602  49.0021
```

Table 2 provides a comparison of actual AMRAAM data from AAMDUS, AAMDUSGC, AAMDUSP, AAMDUSWF and AAMDUSWH to the derived AAMBSLN data. The values are close for the individual subsections except for the variation in WFWT. For the medium range calculation of WFWT, Equation (2.21), only data from two missiles (AMRAAM and SPARROW III) were available. The small sample size has a significant effect on the variation in WFWT.

	AMRAAM	AAMBSLN
WEIGHT	339	362
LENGTH	12	11.7
GCWT	120	112
WHWT	44	45
PWT	154	156
WFWT	21.7	49

Table 2. AMRAAM Versus AAMBSLN MRAAM

What would happen if the user entered a range of 40 NM, rather than the 35 NM range with the same initial length and diameter?

```
» bsln=aambsln(0.6,12,40);
```

Which returns:

```
Initial Estimate
Weight    Length
  lb        ft
351.1754  12.0000
```

Total		Guidance/Control		Warhead	
Weight	Length	Weight	Length	Weight	Length
lb	ft	lb	ft	lb	ft
369.4674	12.2757	120.0697	6.2475	43.7853	1.1680

Propulsion		Wing/Fin
Weight	Length	Weight
lb	ft	lb
156.6104	4.8602	49.0021

Table 3 compares the values for the two missiles. Intuitively, the results do not make sense. If the only parameter changed is the range, one would expect the propulsion section to get heavier and subsequently longer. Why should the guidance subsection weight increase by eight pounds and the propulsion subsection weight stay the same? The variation comes from statistical error in the regression equations used, primarily the small sample size results in insensitivity to some input variable variations. To attempt to use the regression equations to obtain trends in small variations in the input variables may lead to incorrect results. It is often helpful to run AAMBSLN and compare the results to actual missile data found in AAMDUS, etc.

	35 NM MRAAM	40 NM MRAAM
WEIGHT	362.5	369.5
LENGTH	11.7	12.3
GCWT	112.1	120.1
WHWT	44.8	43.8
PWT	156.6	156.6
WFWT	49.0	49

Table 3. Comparison of 35 NM Versus 40 NM MRAAM

The user should remember the purpose of AAMBSLN is to provide a starting point for missile weight and length, not a final, refined value. The numerous interrelated parameters which effect the final configuration of the missile will refine these initial weight estimates.

3. Center of Gravity

One use of the initial subsection weight data is the estimation of the center of gravity of the missile. An estimate of the center of gravity of the missile will be utilized in later Chapters of this thesis. XCGTC1 estimates the center of gravity by using the subsection weights and lengths and assuming that the subsections are point masses with homogeneous densities. The missile configuration is assumed to be such that the guidance/control is subsection first (most forward), followed by the warhead, then by the propulsion subsections (Figure 1). As a first approximation, the weight of the tail is assumed to be 40% of the weight of the wing (Lindsey et al, 1980). Since the tail is located as far aft as possible in a tail-controlled missile, the assumption is made in the program that the tail center of gravity is located at the end of the missile. The difference from actual tail location is small and has small effect on the center of gravity. The center of gravity will be refined later when the tail size and location is better determined. The location of the wing in a tail-controlled missile will be near the body center of gravity and is influenced by the overall center of gravity travel. For the first approximation, the program assumes the wing center of gravity is located at the body center of gravity. The center of gravity along the X axis (Figure 2) measured from the tip of the nose is:

$$l_{CG} = \frac{\sum l_{comp} W_{comp}}{\sum W_{comp}} \tag{2.23}$$

Where l_{comp} is the distance of the center of gravity of a component measured from the nose. The center of gravity along the Y and Z axes is assumed to be zero.

Using the subsection weight and length data from the AMRAAM results in a center of gravity location (measured from the tip of the nose) of:

» lcg = xcgtc1(5.88,120,0.92,44,4.89,154,21.68)

lcg =

6.5591

Which is in units of feet. If the user does not understand how to use XCGTC1, type 'help xcgtc1' at the MATLAB prompt.

4. Moment of Inertia

The moment of inertia about the Y axis can also be calculated from the subsection weights and lengths. The weight of the wing and tail fins can be ignored in the first

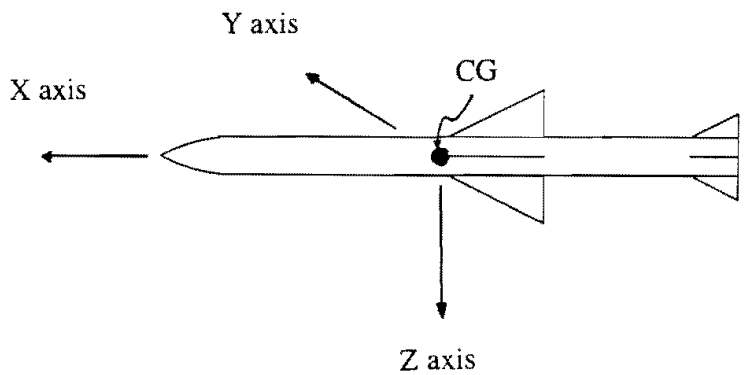


Figure 2. Body Axes

$$I_y = \sum \left(I_{ycomp} + \frac{W_{comp}}{g} x_{comp}^2 \right) \tag{2.24}$$

Where I_{ycomp} is the moment of inertia about the Y axis of a component referenced to the component's center of gravity.

The moment of inertia is calculated using MSLIYY1:

```
» Iy = msliyy1(0.6,5.88,120,0.92,44,4.89,154)
Iy =
  103.9600
```

By symmetry, the moment of inertia about the Z axis is the same as about the Y axis.

5. Script M-File Example

Since the value for the longitudinal center of gravity location just calculated will be used in later chapters, creating a script m-file now will be helpful. The script m-file will allow the retention of a record of the steps used in the design process, and it will prevent the user from reentering commands if a design session is interrupted. It is a good idea to document the reason for each step and the variable definitions to help reduce confusion if the user's design session is interrupted. The following is an example of the script m-file MSLWORK:

```
%******************** MSLWORK ********************
%                          by David A. Ekker
%
% MSLWORK is a script work file used in MSLDSN for designing a
% tail-controlled MRAAM.
```

```
% Calculate the initial center of gravity, lcgi from AMRAAM data.
lcgi = xcgtc1(5.88,120,0.92,44,4.89,154,21.68);
```

The user should now create and save a script m-file similar to the one above before proceeding to the next chapter. Since this is a working file, the contents of the file will change as the user progresses in the design process.

III. AERODYNAMIC LIFT AND STATIC STABILITY

When the aerodynamics of a missile is addressed, one of the major conflicting requirements the designer faces is that of stability versus manueverability. The maneuverability requirement results from small radius turns the missile must perform in order to successfully intercept a target. The faster the missile velocity and the greater the maneuverability of the target, the larger the forces the airframe must generate and sustain. Larger maneuvering forces require larger control forces to provide sufficient static and dynamic stability to control the missile's flight. Since static stability represents the tendency for a body to return to equilibrium, the greater the inherent stability designed into the airframe, the larger the control forces required to overcome them and hence the less maneuverable the missile. Controllability is the ability of the missile to attain and sustain a specified angle of attack. Static stability and controllability are addressed in this chapter. Dynamic stability is addressed in Chapter IV of this thesis.

If the designer is starting the aerodynamic design process without a specific load factor requirement, estimates based on target acceleration capability can be made. One estimate which keeps the miss distance less than 75 feet for most targets comes from Lindsey and Redman (1980).

$$a_m = 3a_T + 10 \tag{3.1}$$

Thus the required missile load factor for a 7g maneuvering target is 31g.

A relative measure of static stability is the static stability margin. The static stability margin is defined as the distance between the center of pressure of the airframe and the center of gravity.

$$\overline{X} = l_{cg} - l_{cp} \tag{3.2}$$

The missile is statically stable as long as $\overline{X}$ is negative. The center of lift of the airframe results from the aerodynamic forces generated by the body, wing and tail. A measure of the maneuverability is the load-factor capability of the missile. The load factor is the ratio of normal force generated divided by the weight of the missile.

$$n = \frac{N}{W} \tag{3.3}$$

The load factor capability results from dividing the load factor by the control deflection. In order to estimate the static stability margin and the load factor capability, estimates of the aerodynamic derivatives $C_{L\alpha}$, $C_{m\alpha}$, $C_{L\delta}$ and $C_{m\delta}$ are required.

For a tail-controlled cruciform arranged missile, the body, wing and tail generate lift and side forces. Due to the symmetry of the cruciform arrangement, if the designer ensures sufficient pitch-control characteristics, satisfactory yaw-control characteristics will result.

In general, the area of the wing is driven by load factor requirements at lower velocities and the wing center of lift is located aft of the center of gravity to help balance the lift of the body. The tail is placed as far aft as possible to increase the length of the tail's moment arm and thus reduce the tail force requirements. Reducing the area of the aerodynamic surfaces helps reduce friction drag resulting in lower propulsion requirements and, correspondingly, reducing missile size and weight. Since the propulsion section of the missile is a large percentage of the total missile weight and is located aft of the missile center of gravity, the center of gravity of the missile shifts forward as the propellant is burned. If the wing is placed to balance the force of the body at the beginning of flight, the forward shift in the center of gravity will result in the wing moment arm lengthening and in a larger static stability margin. The larger static stability margin will require a larger tail force to maintain the same load factor capability. Thus the tail area will depend largely on the missile center of gravity shift and wing position.

The basic methodology for wing and tail sizing starts with estimating the area of the wing based on load factor requirements (WNGSZTC1). An initial estimate of wing placement is made to provide a starting point for analysis (WNGPOS1). The tail area depends on providing enough force to ensure both stability (TLSZSTB1) and controllability (TLSZCN1) throughout the missile's flight. The various flight conditions of the missile need to be checked for tail sizing requirements and then the largest tail area is used. Once the wing and tail area and location are defined, the aerodynamic

derivatives of the missile can be calculated and used to estimate the static margin and the load-factor capability.

The basic procedure of the methodology comes from Lindsey and Redman (1980). The techniques used for estimating aerodynamic derivatives come primarily from Bonney (1950), Chin (1960), Jerger (1961) and DATCOM (Hoak et al, 1978). Procedures found in NACA Technical Report 1307 (Pitts et al, 1953) were used exclusively for lift and center of pressure calculations.

A. NORMAL FORCE VERSUS LIFT FORCE

The relationship between normal, lift and drag forces is:

$$N = L\cos\alpha + D\sin\alpha \tag{3.4}$$

With missile L/D ratios on the order of 5 and angles of attack typically ranging up to 25°, the normal force is approximately equal to the lift force (N = .991L); therefore, the terms will be used synonymously (Lindsey and Redman, 1980). This is an important consideration. If the normal force and lift force were not close in value, an initial drag force estimation would also be required for all wing and tail sizing.

B. WING SIZING

The total lift required by the missile is related to the weight and the load factor required of the missile.

$$L = nW \tag{3.5}$$

$$C_{L\alpha}\alpha_{max}\bar{q}S_{ref} = nW \tag{3.6}$$

Herein, the reference surface area is based on the maximum diameter of the missile body:

$$S_{ref} = \pi\left(\frac{d}{2}\right)^2 \tag{3.7}$$

The required lift curve slope of the missile is:

$$C_{L\alpha} = \frac{nW}{\alpha_{max}\bar{q}S_{ref}} \tag{3.8}$$

The total lift generated by the missile is assumed to be the sum of the lift of the missile's components (Figure 3):

$$L = L_N + (K_{B(W)} + K_{W(B)})L_W + (K_{B(T)} + K_{T(B)})L_T \tag{3.9}$$

Lift due to the body will be addressed as the lift due to the nose since the lift of the body is primarily due to the nose. In terms of dimensionless coefficients:

$$C_L \bar{q} S_{ref} = C_{LN} \bar{q} S_{ref} + (K_{B(W)} + K_{W(B)}) C_{LW} \bar{q} S_W + (K_{B(T)} + K_{T(B)}) C_{LT} \bar{q}_T S_T \tag{3.10}$$

The dynamic pressure at the tail can be related to the dynamic pressure at the wing:

$$\bar{q}_T = \frac{1}{2} \rho M_T^2 a^2 = \frac{M_T^2}{M^2} \bar{q} \tag{3.11}$$

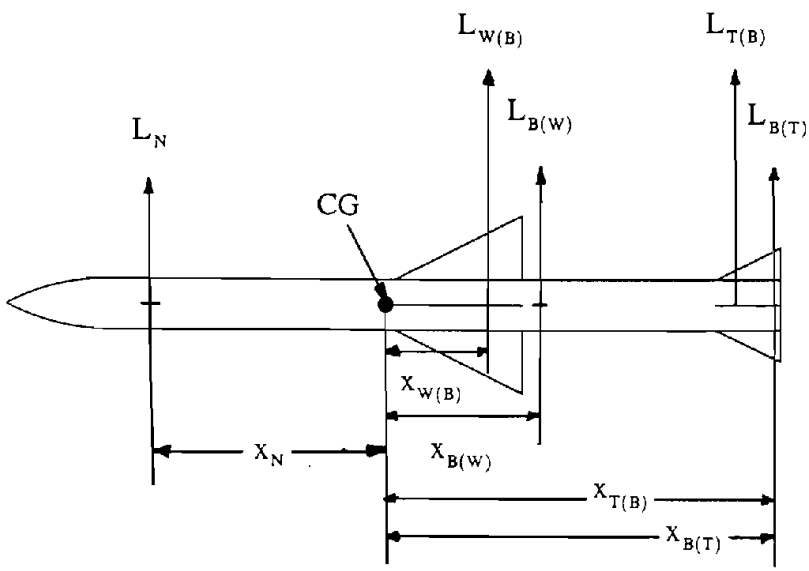


Figure 3. Lift Forces

The flow over the tail tends to be slower due to the effect of the upwind wing interfering with the flow. Thus, the lift coefficient of the missile becomes:

$$C_L = C_{LN} + (K_{B(W)} + K_{W(B)})C_{LW}\frac{S_W}{S_{ref}} + (K_{B(T)} + K_{T(B)})C_{LT}\left(\frac{M_T}{M}\right)^2\frac{S_T}{S_{ref}} \tag{3.12}$$

Differentiating the lift coefficient with respect to angle of attack results in the lift-curve slope:

$$C_{L\alpha} = C_{L\alpha N} + (K_{B(W)} + K_{W(B)})C_{L\alpha W}\frac{S_W}{S_{ref}} + (K_{B(T)} + K_{T(B)})\frac{dC_{LT}}{d\alpha_T}\frac{d\alpha_T}{d\alpha}\left(\frac{M_T}{M}\right)^2\frac{S_T}{S_{ref}} \tag{3.13}$$

The angle of attack seen by the tail differs by a downwash term from what is seen by the nose and wing.

$$\alpha_T = \alpha - \varepsilon \tag{3.14}$$

$$\frac{d\alpha_T}{d\alpha} = 1 - \frac{d\varepsilon}{d\alpha} \tag{3.15}$$

$$\frac{dC_{LT}}{d\alpha_T}\frac{d\alpha_T}{d\alpha} = C_{L\alpha T}\left(1 - \frac{d\varepsilon}{d\alpha}\right) \tag{3.16}$$

$$C_{L\alpha} = C_{L\alpha N} + (K_{B(W)} + K_{W(B)})C_{L\alpha W}\frac{S_W}{S_{ref}} + (K_{B(T)} + K_{T(B)})C_{L\alpha T}\left(1 - \frac{d\varepsilon}{d\alpha}\right)\left(\frac{M_T}{M}\right)^2\frac{S_T}{S_{ref}} \tag{3.17}$$

$$S_W = \frac{(C_{L\alpha} - C_{L\alpha N})S_{ref} + (K_{B(T)} + K_{T(B)})C_{L\alpha T}\left(1 - \frac{d\varepsilon}{d\alpha}\right)\left(\frac{M_T}{M}\right)^2 S_T}{(K_{B(W)} + K_{W(B)})C_{L\alpha W}} \tag{3.18}$$

The lift curve slope of the nose can be estimated using slender body theory and corrected for afterbody extension. The method in DATCOM (Hoak et al, 1978) assumes the length-to-diameter ratio of the body is greater than twenty. Munk's factor $(k_2 - k_1)$ (Jerger, 1961) is added to correct for length-to-diameter ratios less than twenty.

$$C_{L\alpha N} = 2(k_2 - k_1) + c_{dc}\frac{A_p}{S_{ref}}\alpha \tag{3.19}$$

$$A_p = d(l - l_N) + A_{pN} \tag{3.20}$$

$$A_{pN} = l_N\sqrt{R^2 - l_N^2} + R^2\sin^{-1}\left(\frac{l_N}{R}\right) - 2a\,l_N \tag{3.21}$$

$$R = \frac{d}{4} + \frac{l_N^2}{d} \tag{3.22}$$

$$a = R - \frac{d}{2} \tag{3.23}$$

The planform area assumes the nose is approximated by an ogive. The ogive is a good approximation for aerodynamic considerations. Both radar and infrared homing missiles have nose length-to-diameter ratios of approximately two (Lindsey and Redman, 1980). Therefore, the nose length-to-diameter ratio is assumed to be two.

The lift curve slope of the wing is estimated assuming a supersonic $(m\beta \geq 1)$or subsonic $(m\beta < 1)$leading edge. If supersonic:

$$C_{L\alpha W} = \frac{4}{\beta} \tag{3.24}$$

and if subsonic:

$$C_{L\alpha W} = \frac{2\pi \tan\left(\frac{\pi}{2} - \Lambda\right)}{E(k')} \tag{3.25}$$

where $E(k')$ is the complete elliptical integral of the second kind of the modulus k' and:

$$k' = \sqrt{1 - k^2} \tag{3.26}$$

$$k = \sqrt{M^2 - 1}\tan\left(\frac{\pi}{2} - \Lambda\right) \tag{3.27}$$

For supersonic flow:

$$\beta = \sqrt{M^2 - 1} \tag{3.28}$$

For a triangular wing:

$$\tan\left(\frac{\pi}{2} - \Lambda\right) = \frac{A}{4} \tag{3.29}$$

Thus for a triangular planform, the lift curve slope with a subsonic leading edge simplifies to:

$$C_{L\alpha W} = \frac{\pi A}{2\,E\left(\sqrt{1 - \left(\frac{\beta A}{4}\right)^2}\right)} \tag{3.30}$$

The lift curve slope of the tail is similarly estimated using the aspect ratio and local Mach number of the tail. (Bonney, 1950)

The three terms $\frac{d\varepsilon}{d\alpha}$, M_T and S_T are first approximated by:

$$\frac{d\varepsilon}{d\alpha} = 0.5 \tag{3.31}$$

$$M_T = 0.95\, M \tag{3.32}$$

$$S_T = 0.4\, S_w \tag{3.33}$$

the downwash term and the tail surface term will be refined as the wing location and tail area are better known. The estimate for the tail Mach number to wing-body Mach number relationship comes from Jerger (1961) and will be maintained throughout the computations.

The four interference terms require estimates of the wing dimensions. An initial estimate of the wing surface area is required. For an initial estimate, the wing loading is assumed to be 90 lb/ft^2. Then, the initial wing area would be:

$$S_w = \frac{W}{90} \tag{3.34}$$

A triangular planform results in the following dimensions:

$$b_w = \sqrt{\frac{4\, S_w}{\tan \Lambda_w}} \tag{3.35}$$

$$s_w = \frac{1}{2}(d + b_w) \tag{3.36}$$

$$A_w = \frac{b_w^2}{S_w} \tag{3.37}$$

The tail dimensions are similarly computed.

WNGSZTC1 calculates wing surface area and span for a triangular planform by iterating Equation (3.18). WNGSZTC1 requires inputs of missile diameter, length, weight, Mach number, load factor, altitude, leading edge sweep angle and maximum angle of attack. If the user is unsure of the input and output variable definitions, type 'help wngsztc1' at the MATLAB prompt. WNGSZTC1 limits the Mach number from

Mach 1.2 to Mach 5. These limits restrict the calculations to the supersonic regime, where the methodologies used are valid. Using WNGSZTC1 outside this range will result in erroneous results. As a starting assumption, the leading edge sweep angle for the tail is initially assumed to be the same as for the wing. The user should start with a value for leading edge sweep angle based on the historical value from US air-to-air missile data. The maximum angle of attack achieved by the missile is assumed to be 15^{o} if no value for maximum angle of attack is entered.

1. Example One

Using data for the AMRAAM, what would the surface wing area be for a similar tail-controlled missile requiring a load factor of 31 at Mach 1.2 and 10,000 feet? The load factor of 31 is selected from Equation (3.1). A leading edge sweep angle of 55 degrees was selected by reviewing AAMDUSWF and choosing the MRAAM's leading edge sweep angle.

```
» [bw,Sw,amax]=wngsztc1(.6,12,339,1.2,31,10000,55)
bw =
   3.3559
Sw =
   4.0211
amax =
  15.0000
```

As the Mach number increases, the required wing surface area decreases.

```
» [bw,Sw,amax]=wngsztc1(.6,12,339,1.7,31,10000,55)
bw =
   2.4712
```

```
Sw =
   2.1804
amax =
   15.0000
```

At higher Mach numbers it is possible for the body to produce sufficient lift without a wing. WNGSZTC1 calculates the maximum angle of attack required of the body alone to produce the required load factor in this case. For example:

```
» [bw,Sw,amax]=wngsztc1(.6,12,339,4,31,10000,55)
bw =
   0
Sw =
   0
amax =
   11.8653
```

Thus, in this case, the maximum angle of attack returned is the angle of attack of the body alone which will achieve the required load factor.

2. Script M-File

Instead of typing the previous commands at the prompt, the MSLWORK script m-file can be used to run and store the commands. The following commands were added to MSLWORK:

```
d = 0.6;      % Missile diameter
l = 12;       % Missile length
W = 339;      % Missile weight
nm = 31;      % Load Factor
```

```
alt = 10000;    % Altitude
SWP = 55;       % Sweep angle
% Calculate the required wing sizes for Mach 1.2, 1.7, and 4.
M = 1.2
[bw,Sw,amax]=wngsztc1(d,l,W,M,nm,alt,SWP)
M = 1.7
[bw,Sw,amax]=wngsztc1(d,l,W,M,nm,alt,SWP)
M = 4
[bw,Sw,amax]=wngsztc1(d,l,W,M,nm,alt,SWP)
```

3. Example Two

WNGSZTC1 can be used to investigate a number of relationships. WNGSZEX1 uses WNGSZTC1 and the input data from the previous example to calculate wing area versus Mach number while varying the leading edge sweep angle. Figure 4 is the plot which results from running WNGSZEX1. (WNGSZEX1 will take a little over a minute to run on a personal computer (PC) with an Intel 486-33 DX.) The areas of the wings at all three leading edge sweep angles suddenly increase at an increasing rate as Mach 1.2 is approached. The Mach number where this change occurs corresponds to the transition from a supersonic to a subsonic leading edge. Above the transition Mach number, the greater sweep angle results in a slightly smaller wing area. Additionally, the greater sweep angle will result in a lesser drag for the same area. Aerodynamic drag is addressed later. If the user intends the missile to conduct most of its high load factor maneuvers above Mach 1.7, the greater sweep angle would be more efficient.

C. WING POSITION

The position of the wing for a tail-controlled missile affects the tail area. If the wing is positioned to help the tail balance the lifting force of the nose, a smaller tail lifting force is required and thus a smaller tail area. Since the typical missile center of

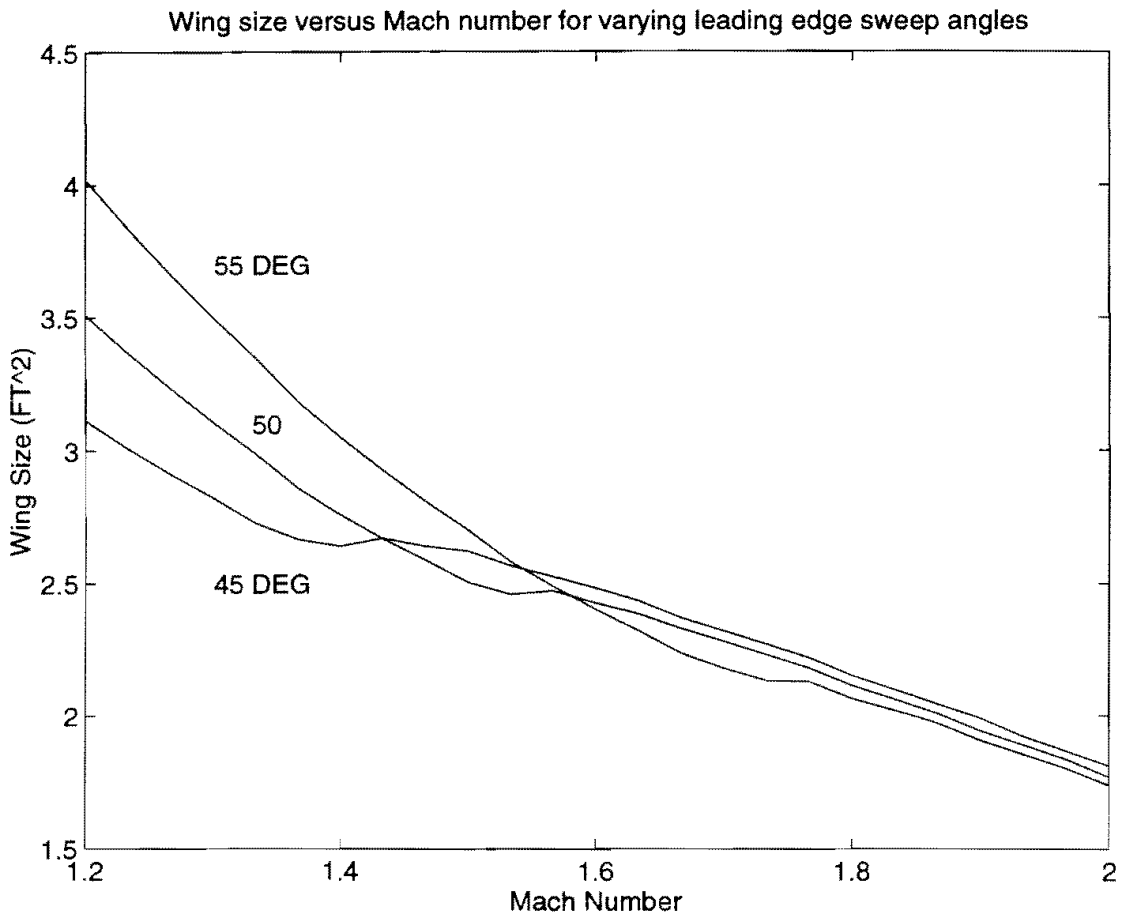


Figure 4. Wing Sizing Example

gravity shifts forward as the fuel is consumed, the length of the moment arms of the nose shortens and the moment arm of the wing lengthens, changing the dynamics of the system. The designer must examine the various flight conditions the missile will encounter and make a decision on wing placement. A good place to initially locate the wing for analysis purposes is to position the wing to balance the nose lift at the initial flight condition. This would result in a neutral stability condition at the beginning of flight. The addition of a tail or a forward shift of the center of gravity would result in a stable condition. Thus, as the fuel is consumed, the center of gravity moves forward and the missile becomes more stable. In later design iterations, the wing will be positioned to minimize tail area and yet still provide sufficient stability and controllability.

MSLDSN will converge to a solution more rapidly if the initial wing placement is relatively close to the final wing position. Thus, MSLDSN starts by calculating the wing position for neutral stability.

The pitching moment equation for the missile is:

$$C_{m\alpha}\alpha\,\bar{q}\,S_{ref}d = C_{L\alpha N}\alpha\,\bar{q}\,S_{ref}x_N - \left(K_{B(W)}x_{B(W)\alpha} + K_{W(B)}x_{W(B)\alpha}\right)C_{L\alpha W}\alpha\,\bar{q}\,S_W - \left(K_{B(T)}x_{B(T)\alpha} + K_{T(B)}x_{T(B)\alpha}\right)\frac{dC_{LT}}{d\alpha_T}\frac{d\alpha_T}{d\alpha}\alpha\,\bar{q}_T\,S_T \tag{3.38}$$

Rearranging and applying the same substitutions for the tail components as was done with the lift equation results in:

$$C_{m\alpha}d = C_{L\alpha N}x_N - \left(K_{B(W)}x_{B(W)\alpha} + K_{W(B)}x_{W(B)\alpha}\right)C_{L\alpha W}\frac{S_W}{S_{ref}} - \left(K_{B(T)}x_{B(T)\alpha} + K_{T(B)}x_{T(B)\alpha}\right)C_{L\alpha T}\left(\frac{M_T}{M}\right)^2\frac{S_T}{S_{ref}}\left(1-\frac{d\varepsilon}{d\alpha}\right) \tag{3.39}$$

The wing will be positioned to exactly balance the lift of the nose without a tail. Thus, the tail component of moment is zero and the total moment is zero for a neutrally stable condition, resulting in:

$$\frac{C_{L\alpha N}x_N S_{ref}}{C_{L\alpha W}S_W} = K_{W(B)}x_{W(B)\alpha} + K_{B(W)}x_{B(W)\alpha} \tag{3.40}$$

The length of a moment arm is the distance from the associated center of lift to the missile center of gravity. Since most distances are referenced to the tip of the nose, the moment arm length is converted to center of lift and center of gravity distances measured from the tip of the nose.

$$x_N = l_{CG} - l_N \tag{3.41}$$

$$x_{B(W)\alpha} = l_{B(W)\alpha} - l_{CG} \tag{3.42}$$

$$x_{W(B)\alpha} = l_{W(B)\alpha} - l_{CG} \tag{3.43}$$

From NACA 1307 (Pitts et al, 1953):

$$l_{B(W)\alpha} = l_W + c_{rW}\left(\frac{x}{c_r}\right)_{B(W)\alpha} \tag{3.44}$$

$$l_{W(B)\alpha} = l_W + c_{rW}\left(\frac{x}{c_r}\right)_{W(B)\alpha} \tag{3.45}$$

And substituting into Equation (3.40):

$$l_W = \frac{\dfrac{C_{L\alpha N}(l_{CG} - l_N)S_{ref}}{C_{L\alpha W}S_W} - \left(K_{W(B)}\left(\dfrac{x}{c_r}\right)_{W(B)\alpha} + K_{B(W)}\left(\dfrac{x}{c_r}\right)_{B(W)\alpha}\right)c_{rW}}{\left(K_{W(B)} + K_{B(W)}\right)} + l_{CG} \tag{3.46}$$

The center of lift of an ogival nose is empirically derived by Miles (Jerger 1960) to be:

$$l_N = \frac{1}{2}\left[\frac{50(M+18)+7M^2P(5M-18)}{40(M+18)+7M^2P(4M-3)}\right] l_N \tag{3.47}$$

$$P = \left(0.083 + \frac{0.096}{M^2}\right)\left(\frac{\sigma}{10}\right)^{1.69} \tag{3.48}$$

$$\sigma = 2\tan^{-1}\left[\frac{d}{2\,l_N}\right]\left(\frac{180^o}{\pi}\right) \tag{3.49}$$

The values resulting from these calculations are best in the 1.5 to 3.5 Mach range and for semi-vertex angles between 10 and 25 degrees. The Miles' formula is used throughout the supersonic flight regime in MSLDSN.

WNGPOS1 returns the distance of the intersection of the body and wing leading edge measured from the tip of the nose. The nose fineness ratio is assumed to be two when calculating the center of pressure of the nose. The value of two is a reasonable value based on historical data (Lindsey and Redman, 1980).

1. Example Three

Run WNGPOS1 using the data for the AMRAAM at Mach 1.2 derived from WNGSZTC1. The center of gravity was estimated using XCGTC1.

```
» lw = wngpos1(0.6,12,339,1.2,6.5591,10000,3.3559,4.0211)
lw =
   5.1980
```

The value is the distance from the tip of the nose to the intersection of the leading edge of the wing and the body. This value is less than the center of gravity position. The user should ask themself if this wing position makes sense? For the given b_W and S_W, the c_r is:

$$c_r = \frac{2\,S_W}{b_W} = 2.40 \tag{3.50}$$

and the center of lift of the wing is approximately:

$$l_W + c_{rW}\left(\frac{x}{c_r}\right)_{W(B)\alpha} \cong 5.2 + 2.4\left(\frac{2}{3}\right) \cong 6.8 \tag{3.51}$$

Therefore the center of pressure of the wing is slightly aft of the center of gravity, as expected.

D. TAIL SIZING

The tail must be sized to provide sufficient stability and controllability throughout the entire flight regime. If the tail is sized for stability at the beginning of flight where the wing placement provides the smallest stability margin, no stability augmentation system will be required. This consideration is important for an air-to-air missile if one wishes to prevent the missile from inadvertently tumbling out of control and impacting the firing aircraft. A stability augmentation system adds weight and cost to the design. The maximum tail area for controllability depends on wing position and center of gravity location.

1. Static Stability

Assume the missile is launched from a subsonic aircraft. The entire subsonic regime will then be checked to ensure static stability.

For subsonic flight, the moment equation becomes:

$$\begin{aligned} C_{m\alpha} d = {} & C_{L\alpha N} x_N - \left(K_{B(W)} x_{B(W)\alpha} + K_{W(B)} x_{W(B)\alpha}\right) C_{L\alpha W} \frac{S_W}{S_{ref}} \\ & - \left(K_{B(T)} x_{B(T)\alpha} + K_{T(B)} x_{T(B)\alpha}\right) \left(\frac{M_T}{M}\right)^2 C_{L\alpha T} \frac{S_T}{S_{ref}} \left(1 - \frac{d\varepsilon}{d\alpha}\right) \end{aligned} \tag{3.52}$$

For the trim condition, the sum of the moments about the missile center of gravity is zero, $\sum M = 0$. Rearranging and solving for tail area results in:

$$S_T = \frac{C_{L\alpha N} x_N S_{ref} - \left(K_{B(W)} x_{B(W)} + K_{W(B)} x_{W(B)}\right) C_{L\alpha W} S_W}{\left(K_{B(T)} x_{B(T)\alpha} + K_{T(B)} x_{T(B)\alpha}\right)\left(\frac{M_T}{M}\right)^2 C_{L\alpha T}\left(1 - \frac{d\varepsilon}{d\alpha}\right)} \tag{3.53}$$

The coefficient of lift for the nose in the subsonic regime comes from slender body theory corrected for the body length-to-diameter ratios less than twenty by Munk's factor:

$$C_{L\alpha N} = 2\left(k_2 - k_1\right) \tag{3.54}$$

The center of pressure for the nose is calculated using slender body theory (Pitts et al, 1953).

$$l_N = l_N - \frac{4\, V_N}{\pi\, d^2} \tag{3.55}$$

For an ogive the volume of the nose is:

$$\frac{V_N}{\pi} = l_N\left(2R^2 - Rd + \frac{d^2}{4}\right) - \frac{l_N^3}{3} - \left(R - \frac{d}{2}\right) l_N \sqrt{R^2 - l_N^2} - \left(R - \frac{d}{2}\right) R^2 \sin^{-1}\frac{l_N}{R} \tag{3.56}$$

$$R = \frac{d}{4} + \frac{l_N^2}{d} \tag{3.57}$$

From DATCOM the wing or tail lift-curve-slope is approximated by:

$$C_{L\alpha} = \frac{2\pi A}{2 + \sqrt{\frac{A^2\beta^2}{\kappa^2}\left(1 + \frac{\tan^2 \Lambda_{c/2}}{\beta^2}\right) + 4}} \tag{3.58}$$

The κ term represents the lift efficiency of the real wing to the ideal flat plate. Thin wings suffer losses in lift-curve from Reynolds-number effects and may also suffer from leading-edge bubble type separation. Thus subsonic flat plates with sharp leading edges will result in a wing efficiency is less than one. Since these various effects are difficult to determine, a conservative value of 0.85 for κ is assumed. This value corresponds to the approximate Reynolds-number effect at $Re = 10^6$. (Hoak et al, 1978)

The $\tan \Lambda_{c/2}$ term can be simplified for a triangular wing to:

$$\Lambda_{c/2} = \tan^{-1} \frac{2}{A} \tag{3.59}$$

The last term to be defined is the rate of change of downwash with respect to angle of attack, $\frac{d\varepsilon}{d\alpha}$. An empirical method will be used from Sanders (1967).

$$\frac{d\varepsilon}{d\alpha} = 4.44\left[K_A K_\lambda K_H \left(\cos \Lambda_{c/4}\right)^{1/2}\right]^{1.19} \tag{3.60}$$

$$K_A = \frac{1}{A_W} - \frac{1}{1 + A_W^{1.7}} \tag{3.61}$$

$$K_\lambda = \frac{10 - 3\lambda_W}{7} \tag{3.62}$$

$$K_H = \frac{1 - \left|\frac{h_H}{b_W}\right|}{\sqrt[3]{\frac{2l_H}{b_W}}} \tag{3.63}$$

With the knowledge that the planform configuration for both the wing and the tail is a triangular (delta) shape and that the wing and tail are in the same plane, the following simplifications can be made:

$$l_H = l_T + \frac{S_T}{b_T} - \left(l_W + \frac{S_W}{b_W}\right) \tag{3.64}$$

$$l_T = 1 - \frac{2\,S_T}{b_T} \tag{3.65}$$

$$l_H = 1 - \frac{S_T}{b_T} - \left(l_W + \frac{S_W}{b_W}\right) \tag{3.66}$$

$$\lambda = 0 \tag{3.67}$$

$$h_H = 0 \tag{3.68}$$

$$\Lambda_{c/4} = \tan^{-1}\left(\frac{3}{A_W}\right) \tag{3.69}$$

The value calculated for $\frac{d\varepsilon}{d\alpha}$ is for incompressible flow. To correct for compressible flow:

$$\left(\frac{d\varepsilon}{d\alpha}\right)_M = \left(\frac{d\varepsilon}{d\alpha}\right)\frac{(C_{L\alpha})_M}{(C_{L\alpha})} \tag{3.70}$$

Where the M subscript implies a value corrected for compressibility. Thus the compressibility corrected value for $\frac{d\varepsilon}{d\alpha}$ is found by multiplying the value of $\frac{d\varepsilon}{d\alpha}$ calculated in Equation (3.60) by the wing lift curve slope at the wing Mach number and dividing by the wing lift curve slope at Mach 0.

TLSZSTB1 solves for the tail area for static stability assuming a subsonic launch (Mach 0 to 0.95). TLSZSTB1 returns the value of the largest tail area required for static stability. Mach 0.95 was chosen as the upper limit which is close to the transonic region. In the transonic region this method is not valid.

a. Example Four

As an example, the data from the Mach 1.2 case of the previous examples is selected. The tail leading edge sweep angle can be varied to investigate the variation in tail area for stability with respect to sweep angle. The user should try a 55° leading edge sweep first.

```
» [bt,St,lt]=tlszstb1(0.6,12,6.5591,3.3559,4.0211,5.1980,55)
bt =
   1.0456
St =
   0.4630
lt =
  11.2534
```

A 45° leading edge sweep angle results in a smaller tail area.

```
» [bt,St,lt]=tlszstb1(0.6,12,6.5591,3.3559,4.0211,5.1980,45)
bt =
   1.0573
St =
   0.3741
lt =
  11.4713
```

2. Controllability

For the controllability condition, the ability of the tail to provide sufficient force to hold the missile at angle of attack is calculated. The moment equation of the missile at angle of attack with the moment due to tail deflection is, in coefficient form:

$$C_m = C_{m\alpha}\alpha + C_{m\delta}\delta \tag{3.71}$$

The moment term for the tail deflection is:

$$C_{m\delta}\delta\,\bar{q}\,S_{ref}\,d = -\left(k_{T(B)}x_{T(B)\delta} + k_{B(T)}x_{B(T)\delta}\right)C_{L\alpha T}S_T\delta\,\bar{q}_T \tag{3.72}$$

Simplifying:

$$C_{m\delta}\,d = -\left(k_{T(B)}x_{T(B)\delta} + k_{B(T)}x_{B(T)\delta}\right)\left(\frac{M_T}{M}\right)^2\frac{S_T}{S_{ref}}C_{L\alpha T} \tag{3.73}$$

Setting $\sum M = 0$ and solving for the tail planform area:

$$S_T = \frac{\left(\frac{M}{M_T}\right)^2\left[C_{L\alpha N}x_N S_{ref} - \left(K_{B(W)}x_{B(W)\alpha} + K_{W(B)}x_{W(B)\alpha}\right)C_{L\alpha W}S_W\right]\alpha_{max}}{C_{L\alpha T}\left[\left(K_{B(T)}x_{B(T)\alpha} + K_{T(B)}x_{T(B)\alpha}\right)\left(1 - \frac{d\varepsilon}{d\alpha}\right)\alpha_{max} + \left(k_{B(T)}x_{B(T)\delta} + k_{T(B)}x_{T(B)\delta}\right)\delta_{max}\right]} \tag{3.74}$$

As with the wing, the lengths of the moment arms for the tail are:

$$x_{B(T)\alpha\text{ or }\delta} = l_{B(T)\alpha\text{ or }\delta} - l_{CG} \tag{3.75}$$

$$x_{T(B)\alpha\text{ or }\delta} = l_{T(B)\alpha\text{ or }\delta} - l_{CG} \tag{3.76}$$

The center of lift at the tail can be found in a manner similar to that of the wing.

$$l_{B(T)\alpha \text{ or } \delta} = l_T + c_{rT}\left(\frac{x}{c_r}\right)_{B(T)\alpha \text{ or } \delta} \tag{3.77}$$

$$l_{T(B)\alpha \text{ or } \delta} = l_T + c_{rT}\left(\frac{x}{c_r}\right)_{T(B)\alpha \text{ or } \delta} \tag{3.78}$$

The location of the intersection of leading edge of the tail and the body is simply the length of the missile minus the root chord of the tail:

$$l_T = l - c_{rT} \tag{3.79}$$

The downwash term is calculated for the supersonic case using the method of Lomax and Sluder (Jerger, 1961).

TLSZCN1 calculates the tail area with additional inputs of maximum angle of attack and maximum deflection angle. If no inputs for angle of attack or deflection angle are given, default values of 15° are assumed.

a. Example Five

Continuing with the data from Mach 1.2, the area and location of the tail is quickly calculated.

```
» [bt,St,lt]=tlszcn1(0.6,12,6.5591,1.2,3.3559,4.0211,5.1980,55)
bt =
   1.0996
St =
   0.3926
lt =
  11.2859
```

Changing the tail leading edge sweep angle to 45° again results in a smaller tail area.

```
» [bt,St,lt]=tlszcn1(0.6,12,6.5591,1.2,3.3559,4.0211,5.1980,45)
bt =
   1.1573
St =
   0.2893
lt =
  11.5000
```

Comparing the results of tail area for controllability $\left(S_T = 0.3926 \text{ ft}^2\right)$ with the results for stability $\left(S_T = 0.4630 \text{ ft}^2\right)$ from example four, we see the tail area for stability $\left(S_T = 0.4630 \text{ ft}^2\right)$ is larger and is the determining value for this case.

b. Example Six

The center of gravity of the missile shifts forward as the solid propellant is consumed. For a first check of center of gravity movement over the missile flight path, the user should rerun XCGTC1 with the propulsion subsection weight at the end of burn. If the value is not yet known, assume the propulsion subsection weight at the end of the burn is on the order of 10% of the original weight; 90% of the subsection weight is due to the propellant.

```
» lcgf=xcgtc1(5.88,120,0.92,44,4.89,0.1*154,21.68)
lcgf =
   4.5423
```

If the user reruns TLSZCN1 for the most forward center of gravity shift:

```
» [bt,St,lt]=tlszcn1(0.6,12,4.5423,1.2,3.3559,4.0211,5.1980,55)
bt =
   2.1260
```

```
St =
   0.7591
lt =
   11.2859
```

The user should note that the tail area required for controllability $\left(S_T = 0.7591 \text{ ft}^2\right)$ at the end of flight is larger than the tail area required for stability $\left(S_T = 0.4630 \text{ ft}^2\right)$ at the beginning of flight and is now the determining case.

The optimum wing position and tail size combination is the wing location which results in the smallest tail area for the various missile flight conditions. To obtain the optimum tail area, various wing locations and stability and controllability conditions need to be checked. TLSZEX1 is an example script m-file which illustrates the checking of five flight conditions and returns the optimum wing location-tail area combination. The conditions are:

- Launch stability
- Mach 1.2, 15° angle of attack maneuver at the beginning of flight
- Mach 1.2, 15° angle of attack maneuver at the end of flight
- Mach 1.7, 15° angle of attack maneuver at the beginning of flight
- Mach 1.7, 15° angle of attack maneuver at the end of flight

The wing designed for the Mach 1.7 condition of Example One $\left(S_W = 2.1804 \text{ ft}^2\right)$ is used. (TLSZEX1 takes approximately five minutes to run on a PC with an Intel 486-33 DX. The number of iterations counts down and is displayed to the screen along with the total elapsed time. The execution of TLSZEX1 can be stopped by pressing "Control-Break.") Executing TLSZEX1 results in Figures 5 and 6 and the following:

```
» tlszex1
btopt =
   1.3150
```

Stopt =

0.6174

lwopt =

5.6824

Each of the lines in Figure 5 represents one of the four controllability conditions considered and the launch stability. TLSZEX1 selects the wing location where the most limiting tail area is the smallest. The actual values returned by TLSZEX1 is calculated from the individual data points and not by the actual intersection point of the lines. If higher precision is desired, reduce the wing location range and/or increase the number of data points calculated in TLSZEX1. If the lower value for wing location is chosen as 5.5 and the upper value as 6.5 in TLSZEX1, the following values are returned:

btopt =

1.2988

Stopt =

0.6023

lwopt =

5.7083

E. INITIAL ESTIMATE REFINEMENT

The values calculated for the wing and tail by WNGSZTC1 assumed the tail area was $0.4S_W$ $\left(S_T = 0.8722 \text{ ft}^2\right)$. The value of the tail area has been refined by running TLSZEX1 $\left(S_T = 0.6023 \text{ ft}^2\right)$ and the original assumption of WNGSZTC1 is no longer valid. WNGSZTC1 has the option to accept values for the actual tail dimensions and wing location, and then recalculate the wing area with this new tail area. Additionally, the center of gravity calculations assumed tail and wing locations. With actual wing and

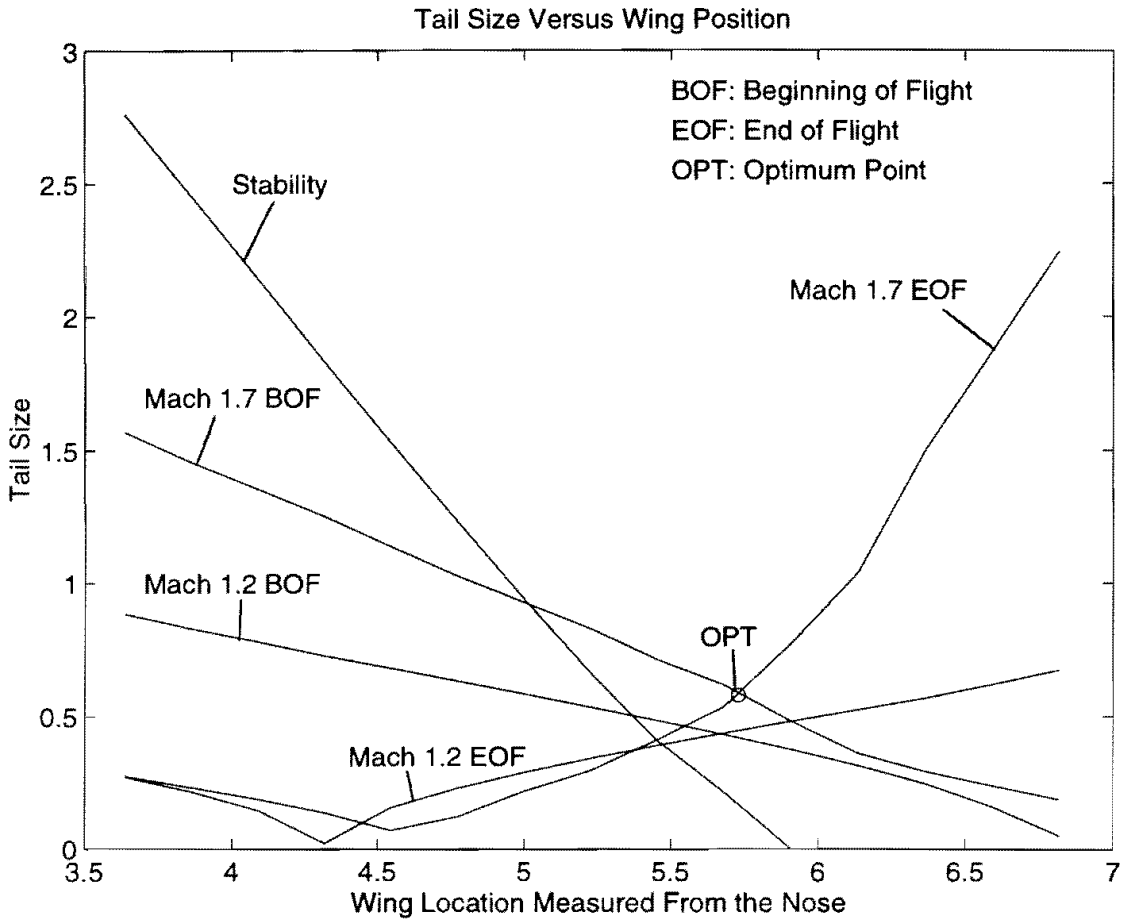


Figure 5. Optimizing Tail Area and Wing Location

tail locations known, a better center of gravity calculation can be made. XCGTC2 is used to calculate this revised center of gravity.

TLSZEX2 uses the values for tail span, area and wing locations from the last example and recalculates tail and wing dimensions. From Figure 5, the stability and Mach 1.2 controllability conditions are known not to be the determining cases and are thus removed from the calculations. Additionally, TLSZEX2 returns the difference in wing area (dSw) and tail area (dSt) from the last example, to help the user gauge how large an effect the refinement has on the previous wing and tail areas.

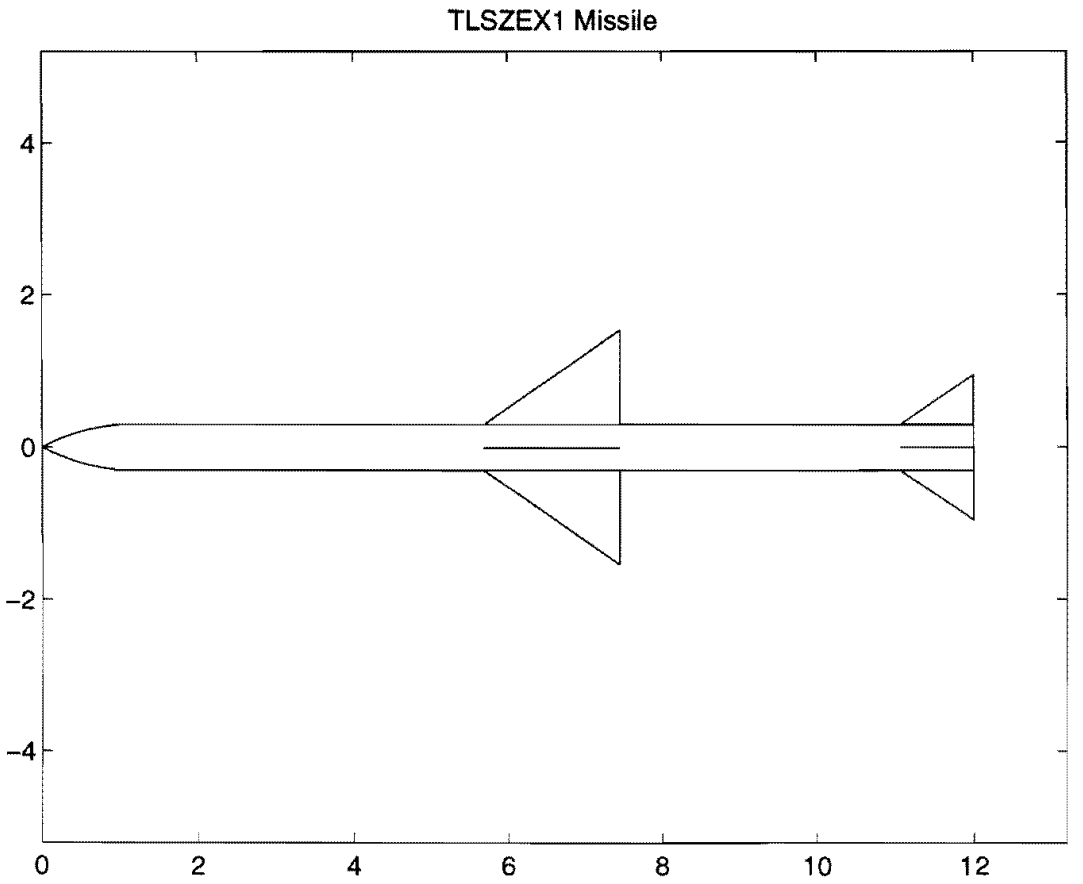


Figure 6. Missile Configured in TLSZEX1

```
» tlszex2
dSw =
   0.1994
dSt =
   0.0035
bwopt =
   2.5817
Swopt =
   2.3798
```

```
lwopt =
   5.5767
btopt =
   1.2950
Stopt =
   0.5988
```

Since the changes in wing and tail areas as a result of running TLSZEX2 are small, no further iterations are necessary. If the user wants a more precise answer, making the wing position increment smaller in TLSZEX2 should be tried first, before another iteration attempt is made.

F. STATIC STABILITY MARGIN AND LOAD-FACTOR CAPABILITY

With the external configuration of the missile defined, the wing-body-tail lift-curve slope, pitching-moment-curve slope for angle of attack and pitching-moment-curve slope for tail incidence angle coefficients can be calculated using Equations (3.17), (3.39) and (3.73), respectively. CLAWBT, CMAWBT and CMDWBT are the associated m-files. The values calculated are valid for the subsonic and supersonic Mach regions. The region between Mach 0.95 to 1.2 is considered transonic and no coefficient values are calculated in this region. Transonic flow interaction is beyond the capabilities of MSLDSN and is not considered.

The lift-curve slope for tail deflection is:

$$C_{L\delta} = C_{L\alpha T}\left(\frac{M_T}{M}\right)^2 \left(k_{B(T)} + k_{T(B)}\right)\frac{S_T}{S_{ref}} \tag{3.80}$$

The m-file which generates values for $C_{L\delta}$ is CLDWBT.

With the necessary aerodynamic coefficients known, the static stability margin and the load-factor capability are easily calculated. The derivation of both parameters comes from Chin (1961).

1. Static Stability Margin

The static stability margin is derived from the pitching moment equation about the center of gravity for the complete missile:

$$M = L\overline{X} \tag{3.81}$$

where:

$$\overline{X} = l_{cg} - l_{cp} \tag{3.82}$$

The missile is stable as long as $\overline{X}$ is negative. Dividing Equation (3.81) by $\overline{q}S_{ref}d$ results in the moment equation in terms of moment and lift coefficients.

$$C_m = C_L \frac{\overline{X}}{d} \tag{3.83}$$

Differentiating with respect to angle of attack and rearranging terms gives the static margin in terms of fractions of missile diameter.

$$\frac{\overline{X}}{d} = \frac{C_{m\alpha}}{C_{L\alpha}} \tag{3.84}$$

2. Load-Factor Capability

Derivation of the load-factor capability starts with the definition of the load factor.

$$n = \frac{N}{W} = C_{LTR} \frac{\overline{q}S_{ref}}{W} \tag{3.85}$$

$$C_{LTR} = \frac{nW}{\overline{q}S_{ref}} \tag{3.86}$$

For the trimmed condition, the moment equation simplifies to:

$$-C_{m\alpha}\alpha_{TR} = C_{m\delta}\delta_{TR} \tag{3.87}$$

$$\alpha_{TR} = -\frac{C_{m\delta}}{C_{m\alpha}}\delta_{TR} \tag{3.88}$$

The lift at trim resulting from the trim angle of attack and tail deflection angle is:

$$C_{LTR} = C_{L\alpha}\alpha_{TR} + C_{L\delta}\delta_{TR} \tag{3.89}$$

$$C_{LTR} = C_{L\alpha}\left(-\frac{C_{m\delta}\delta_{TR}}{C_{m\alpha}}\right) + C_{L\delta}\delta_{TR} \tag{3.90}$$

Substituting Equation (3.86) for lift at trim and rearranging to obtain the load factor per unit control deflection is:

$$\frac{nW}{\bar{q}S_{ref}} = \left(-\frac{C_{m\delta}}{C_{m\alpha}} C_{L\alpha} + C_{L\delta} \right) \delta_{TR} \tag{3.91}$$

$$\frac{n}{\delta} = \left[C_{L\delta} - C_{L\alpha} \frac{C_{m\delta}}{C_{m\alpha}} \right] \frac{\bar{q}S_{ref}}{W} \tag{3.92}$$

The user should run SMANDLF for an example of calculating the static stability margin and load-factor capability.

IV. DYNAMIC CHARACTERISTICS

The dynamic characteristics are extremely important for determining the rapidity with which the airframe of the missile will respond to command inputs. For a cruciform missile design a two-degree-of-freedom model is sufficient to evaluate the response characteristics of the airframe for pitch or yaw. The details of deriving the full six-degree-of-freedom equations of motion can be found in many texts (e.g. Etkin (1972)) and will not be covered here. The derivation of the two-degree-of-freedom model comes from Nielsen (1960) with the exception that simplifications are not made until the final steps of the derivation. The equations of motion used for the longitudinal case are the Z axis force equation and the Y axis moment equation in wind axes coordinates (Figure 7).

$$\sum F_z = -mV\dot{\gamma} \tag{4.1}$$

$$\sum M_y = \dot{q}\, I_y = \ddot{\theta}\, I_y \tag{4.2}$$

The sum of the lift forces results from the lift of the entire missile and the lift of the tail at incidence.

$$\sum F_z = -L_\alpha \alpha - L_\delta \delta \tag{4.3}$$

Converting lift forces to aerodynamic coefficients:

$$\sum F_z = -\bar{q}\, S_{ref} C_{L\alpha} \alpha - \bar{q}\, S_{ref} C_{L\delta} \delta = -mV\dot{\gamma} \tag{4.4}$$

Rearranging the equation and solving in terms of the flight path angle:

$$\dot{\gamma} = \frac{\bar{q} S_{ref}}{mV} C_{L\alpha} \alpha + \frac{\bar{q} S_{ref}}{mV} C_{L\delta} \delta \tag{4.5}$$

The flight path angle is equal to the pitch angle minus the angle of attack.

$$\gamma = \theta - \alpha \tag{4.6}$$

$$\dot{\gamma} = \dot{\theta} - \dot{\alpha} \tag{4.7}$$

Solving Equation (4.5) in terms of elevation angle rate:

$$\dot{\theta} = \dot{\alpha} + \frac{\bar{q} S_{ref}}{mV} C_{L\alpha} \alpha + \frac{\bar{q} S_{ref}}{mV} C_{L\delta} \delta \tag{4.8}$$

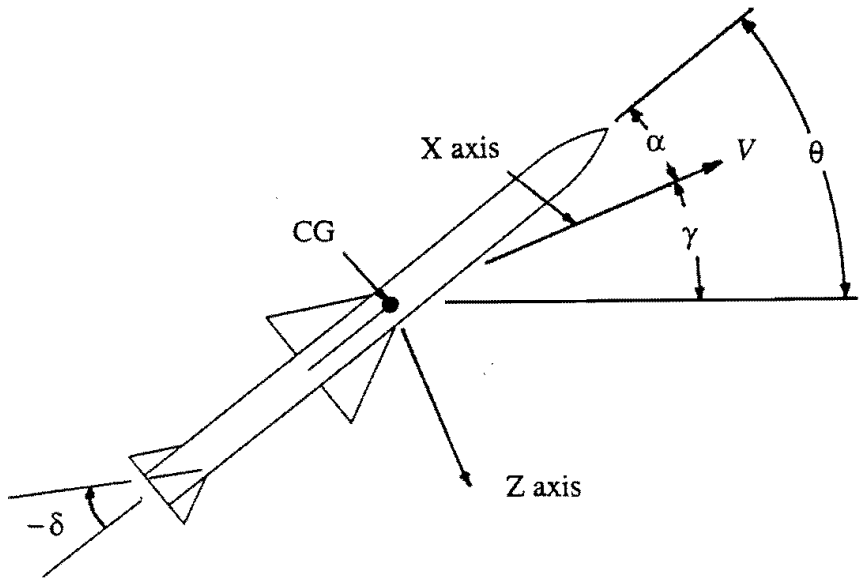


Figure 7. Flight Path and Elevation Angle Relationship in Wind Axes

The elevation angle acceleration is the derivative with respect to time of the elevation angle rate:

$$\ddot{\theta} = \ddot{\alpha} + \frac{\bar{q}S_{ref}}{mV} C_{L\alpha} \dot{\alpha} + \frac{\bar{q}S_{ref}}{mV} C_{L\delta} \dot{\delta} \tag{4.9}$$

The moment equation about the Y axis in terms of the individual moments is:

$$\sum M_y = I_y \ddot{\theta} = M_\alpha \alpha + M_{\dot{\alpha}} \dot{\alpha} + M_{\dot{\theta}} \dot{\theta} + M_\delta \delta + M_{\dot{\delta}} \dot{\delta} \tag{4.10}$$

Converting moments to aerodynamic derivatives results in:

$$\sum M_y = I_y \ddot{\theta} = \bar{q} S_{ref} d \left[C_{m\alpha} \alpha + C_{m\delta} \delta + \frac{d}{2V} \left(C_{m\dot{\alpha}} \dot{\alpha} + C_{m\dot{\theta}} \dot{\theta} + C_{m\dot{\delta}} \dot{\delta} \right) \right] \tag{4.11}$$

Rearranging and solving in terms of the elevation angle acceleration:

$$\ddot{\theta} - \frac{\bar{q}S_{ref}d}{I_y}\left[\frac{d}{2V}\left(C_{m\dot{\theta}}\dot{\theta} + C_{m\dot{\alpha}}\dot{\alpha}\right) + C_{m\alpha}\alpha\right] = \frac{\bar{q}S_{ref}d}{I_y}\left[\frac{d}{2V}C_{m\dot{\delta}}\dot{\delta} + C_{m\delta}\delta\right] \tag{4.12}$$

Substituting Equations (4.8) and (4.9) for the elevation angle rate and acceleration:

$$\begin{aligned}\ddot{\alpha} + \frac{\bar{q}S_{ref}}{mV}\left(C_{L\alpha}\dot{\alpha} + C_{L\delta}\dot{\delta}\right) - C_{m\dot{\theta}}\frac{d}{2V}\frac{\bar{q}S_{ref}d}{I_y}\left[\ddot{\alpha} + \frac{\bar{q}S_{ref}}{mV}\left(C_{L\alpha}\alpha + C_{L\delta}\delta\right)\right] \\ - \frac{\bar{q}S_{ref}d}{I_y}\left[\frac{d}{2V}C_{m\alpha}\dot{\alpha} + C_{m\alpha}\alpha\right] = \frac{\bar{q}S_{ref}d}{I_y}\left[C_{m\delta}\delta + \frac{d}{2V}C_{m\dot{\delta}}\dot{\delta}\right]\end{aligned} \tag{4.13}$$

The elevation angle rate is equal to the pitch rate; this is simply a transformation from one reference frame to another.

$$\dot{\theta} = q \tag{4.14}$$

Similarly:

$$C_{m\dot{\theta}} = C_{mq} \tag{4.15}$$

Making this substitution and grouping terms results in:

$$\begin{aligned}\ddot{\alpha} + \left[\frac{\bar{q}S_{ref}}{mV}C_{L\alpha} - \frac{\bar{q}S_{ref}d}{I_y}\frac{d}{2V}\left(C_{mq} + C_{m\dot{\alpha}}\right)\right]\dot{\alpha} - \frac{\bar{q}S_{ref}d}{I_y}\left[\frac{\bar{q}S_{ref}}{mV}\frac{d}{2V}C_{mq}C_{L\alpha} + C_{m\alpha}\right]\alpha = \\ \frac{\bar{q}S_{ref}d}{I_y}\left[C_{m\delta} + \frac{\bar{q}S_{ref}}{mV}C_{L\delta}\right]\delta + \left[\frac{\bar{q}S_{ref}d}{I_y}\frac{d}{2V}C_{m\dot{\delta}} - \frac{\bar{q}S_{ref}}{mV}C_{L\dot{\delta}}\right]\dot{\delta}\end{aligned} \tag{4.16}$$

The magnitude of the $\dot{\delta}$ terms are generally small compared to the δ terms and it is assumed that these terms can be ignored.

Equation (4.16) can be placed into the following form:

$$\ddot{\alpha} + 2\zeta\omega_n\dot{\alpha} + \omega_n^2\alpha = K\delta(t) \tag{4.17}$$

The left hand side of Equation (4.17) is the characteristic equation of the system. The roots of this equation determine the type of response the system. The two roots of the characteristic equation can result from one of five possible combinations:

- Two positive real roots
- Two negative real roots
- One positive and one negative real root
- Complex conjugate roots with positive real parts
- Complex conjugate roots with negative real parts

The dynamic stability of the system is dependent on the real parts of the roots. If the real parts of the roots are positive, the system response will be divergent and dynamically unstable. If the real parts of the roots are negative, the system response will be convergent and dynamically stable. If there are complex roots, oscillatory motion of the system will result. (Ogata, 1990)

From Equation (4.16), the values of the coefficents are:

$$\omega_n^2 = -\frac{\bar{q}S_{ref}d}{I_y}\left[\frac{\bar{q}S_{ref}}{mV}\frac{d}{2V}C_{mq}C_{L\alpha} + C_{m\alpha}\right] \tag{4.18}$$

$$\zeta = \frac{\left[\frac{\bar{q}S_{ref}}{mV}C_{L\alpha} - \frac{\bar{q}S_{ref}d}{I_y}\frac{d}{2V}\left(C_{mq} + C_{m\dot{\alpha}}\right)\right]}{2\omega_n} \tag{4.19}$$

$$K = \frac{\bar{q}S_{ref}d}{I_y}\left[\frac{d}{2V}C_{m\delta} + \frac{\bar{q}S_{ref}}{mV}C_{L\delta}\right] \tag{4.20}$$

As $t \to \infty$, $\ddot{\alpha}(\infty) \to 0$ and $\dot{\alpha}(\infty) \to 0$. Thus the steady state angle of attack of a dynamically stable system can be related to the deflection angle of the tail.

$$\alpha(\infty) = \frac{K\delta(\infty)}{\omega_n^2} \tag{4.21}$$

Equation (4.16) can be solved using Laplace transforms or other techniques. Assuming $\alpha(0) = 0$ and $\dot{\alpha}(0) = 0$, the solution for the underdamped ($\zeta < 1$) case:

$$\alpha(t) = \alpha(\infty)\left[1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\cos(\omega_d t + \phi)\right] \tag{4.22}$$

$$\phi = \tan^{-1}\left(\frac{\zeta}{\sqrt{1-\zeta^2}}\right) \tag{4.23}$$

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{4.24}$$

For the critically damped case ($\zeta = 1$):

$$\alpha(t) = \alpha(\infty)\left[1 - e^{-\zeta\omega_n t}\right] \tag{4.25}$$

For the overdamped case ($\zeta > 1$):

$$\alpha(t) = \alpha_{final}\left[1 - e^{-\zeta\omega_n t} + \frac{\zeta}{\sqrt{\zeta^2 - 1}} e^{-\zeta\omega_n t}\frac{e^{-\omega_d t} - e^{\omega_d t}}{2}\right] \tag{4.26}$$

In the underdamped case, the time for the first peak, t_p, to occur and the maximum value of the overshoot, M_p, at t_p are important considerations.

$$t_p = \frac{\pi}{\omega_d} \tag{4.27}$$

$$M_p = \frac{\alpha(t_p) - \alpha(\infty)}{\alpha(\infty)} = e^{-(\zeta/\sqrt{1-\zeta^2})\pi} \tag{4.28}$$

The terms $C_{m\dot{\alpha}}$ and C_{mq} are the only unknowns in the equations. The derivations of these coefficients are from Chin (1961).

A. $M_{\dot{\alpha}}$

$M_{\dot{\alpha}}$ results from the finite time required for the wing downwash to arrive at the tail (Figure 8). The downwash in this case is:

$$\varepsilon = \frac{d\varepsilon}{d\alpha}\left(\alpha - \frac{d\alpha}{dt}\Delta t\right) \tag{4.29}$$

$$\Delta t = \frac{x_{W\to T}}{V} \tag{4.30}$$

Δt is the time it takes for the downwash to travel the distance from the wing to the tail, $x_{W\to T}$. The moment equation for the tail is:

$$M_T = -C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\alpha_T \bar{q}_T S_T \tag{4.31}$$

The angle of attack as seen by the tail is:

$$\alpha_T = \alpha - \varepsilon \tag{4.32}$$

Substituting the angle of attack and the downwash into the moment equation:

$$M_T = -C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\left[\alpha - \frac{d\varepsilon}{d\alpha}\left(\alpha - \frac{d\alpha}{dt}\frac{x_{W\to T}}{V}\right)\right]\bar{q}_T S_T \tag{4.33}$$

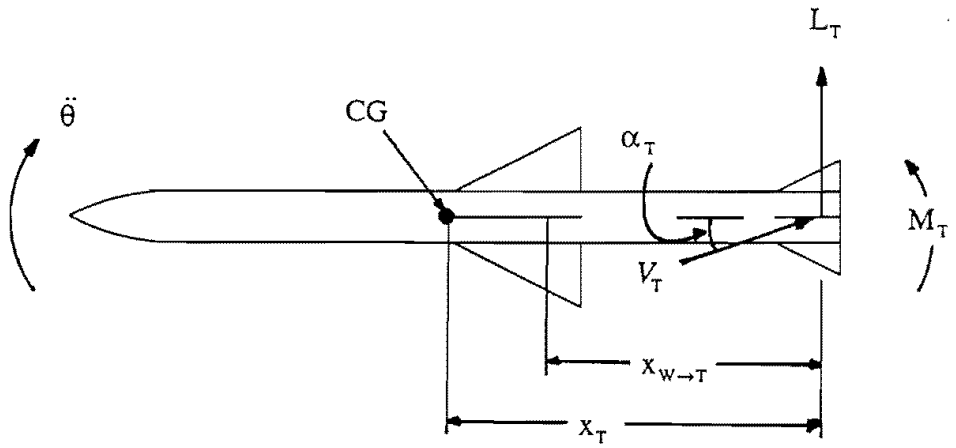


Figure 8. Tail Moment

Dividing by $\bar{q}S_{ref}d$ results in:

$$C_{mT} = -C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\left[\alpha - \frac{d\varepsilon}{d\alpha}\left(\alpha - \dot{\alpha}\frac{x_{W\to T}}{V}\right)\right]\frac{\bar{q}_T S_T}{\bar{q}S_{ref}d} \tag{4.34}$$

The definition of $C_{m\dot{\alpha}}$:

$$C_{m\dot{\alpha}} = \frac{\partial C_m}{\partial\left(\frac{\dot{\alpha}d}{2V}\right)} \tag{4.35}$$

Due to this definition, the derivation of $C_{m\dot{\alpha}}$ for missiles is different than for aircraft. The aircraft derivation uses the wing mean aerodynamic chord vice the body diameter for the reference length. The missile definition causes higher values for the coefficient compared

to the aircraft definition. The user should remember this term is multiplied by $\frac{\bar{q}d^2S_{ref}}{2V}$ to get a moment. The desired aerodynamic derivative is:

$$C_{m\dot{\alpha}} = -2C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\frac{S_T x_{W\to T}}{S_{ref}d^2}\left(\frac{M_T}{M}\right)^2\frac{d\varepsilon}{d\alpha} \tag{4.36}$$

$$x_{W\to T} = l_T + \left(\frac{x}{c_r}\right)_{T(B)}c_{rT} - \left(l_W + \left(\frac{x}{c_r}\right)_{W(B)}c_{rW}\right) \tag{4.37}$$

B. M_q

M_q is the result of the angle of attack change seen by an aerodynamic surface due to the rotation of the missile about its center of gravity. The tail is the largest contributor for most missile configurations due to the length of the tail's moment arm, although the nose and the wing do contribute. The damping due to the tail is derived from:

$$\Delta M_T = -C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\Delta\alpha_T\bar{q}_T S_T \tag{4.38}$$

Dividing by $\bar{q}S_{ref}d$:

$$\Delta C_{mT} = -C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\Delta\alpha_T\frac{\bar{q}_T S_T}{\bar{q}S_{ref}} \tag{4.39}$$

$$\Delta\alpha_T = \tan^{-1}\left(\frac{q\,x_{T(B)}}{V}\right) \cong \frac{q\,x_{T(B)}}{V} \tag{4.40}$$

$$\Delta C_{mT} = -C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\frac{q}{V}\frac{S_T}{S_{ref}}\left(\frac{M_T}{M}\right)^2\frac{x_{T(B)}}{d} \tag{4.41}$$

$$C_{mq} = \frac{\partial C_m}{\partial\left(\frac{qd}{2V}\right)} \tag{4.42}$$

$$C_{mqT} = -2C_{L\alpha T}\left[K_{B(T)}x_{B(T)} + K_{T(B)}x_{T(B)}\right]\frac{S_T}{S_{ref}}\left(\frac{M_T}{M}\right)^2\frac{x_{T(B)}}{d^2} \tag{4.43}$$

The damping terms due to the nose and the wing are similarly derived.

$$C_{mq} = -\frac{2}{d^2}\left\{\begin{array}{l} C_{L\alpha N}X_N^2 + C_{L\alpha W}\left[K_{B(W)}X_{B(W)} + K_{W(B)}X_{W(B)}\right]\dfrac{S_W}{S_{ref}}X_{W(B)} \\ +C_{L\alpha T}\left[K_{B(T)}X_{B(T)} + K_{T(B)}X_{T(B)}\right]\dfrac{S_T}{S_{ref}}\left(\dfrac{M_T}{M}\right)^2 X_{T(B)} \end{array}\right\} \quad (4.44)$$

C. EXAMPLE

MSLRSPNS calculates the natural frequency, damping factor, roots of the missile system, the final angle of attack and plots the missile response for a unit step tail deflection. If the airframe is underdamped, MSLRSPNS also returns the time for the first peak to occur and the maximum overshoot. Since the missile is tail-controlled, the default step value for the deflection in MSLRSPNS is negative. The script file MSLREX1 is used to store the parameters of the missile designed at the end of Chapter III and to run MSLRSPNS. Figure 9 is the resulting plot.

```
» mslrex1
```

Returns:

```
wn =
  16.4515
z =
   0.0964
p =
 -1.5865 +16.3749i
 -1.5865 -16.3749i
af =
  22.0291
tp =
   0.1919
```

Mp =

0.7376

The values of the real parts of the poles in this example are negative, thus the system is stable. The fact that the roots are complex implies the system is oscillatory as evidenced in Figure 9. The system eventually converges to a steady state angle of attack of 22.0291 degrees. The low damping results in a high overshoot. The high overshoot could cause excessive loading of the airframe or flight control problems.

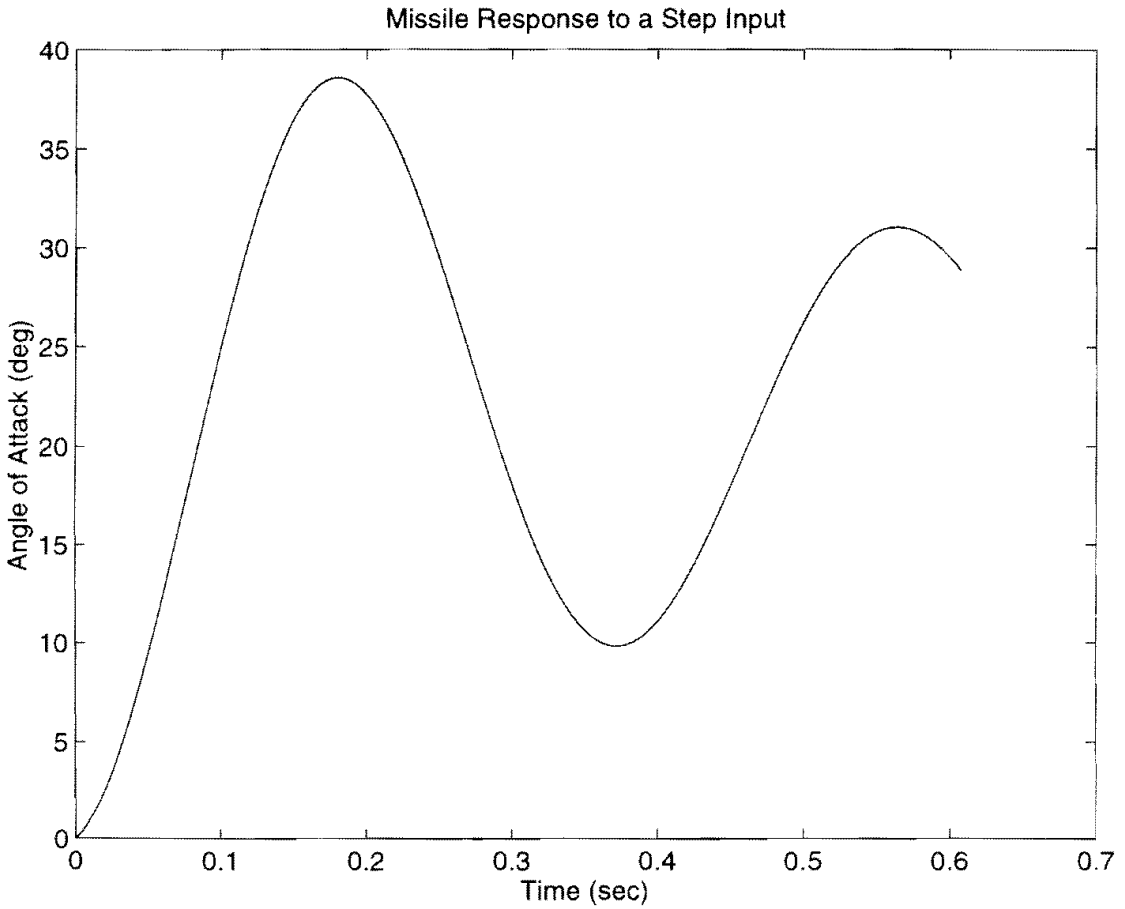


Figure 9. Missile Response to Unit Step Tail Deflection

V. DRAG

The total drag of a missile is composed of the zero lift drag, the drag due to lift and the drag due to control surface deflection.

$$D = D_0 + D_L + D_{TR} \tag{5.1}$$

$$D = (C_{D0} + C_{DL} + C_{DTR})\bar{q}S_{ref} \tag{5.2}$$

Drag values are required for the sizing of the propulsion system and for an estimation of the missile's performance.

A. ZERO LIFT DRAG

In many cases, the value for the zero lift drag of the wing-body-tail combination can be used for initial performance and propulsion calculations. The zero lift drag coefficient for the missile is calculated by summing the components of the zero lift drag coefficients of the individual components.

$$C_{D0} = C_{D0B} + 2C_{D0W}\frac{S_W}{S_{ref}} + 2C_{D0T}\frac{S_T}{S_{ref}} \tag{5.3}$$

The zero lift drag of each aerodynamic component is composed of a friction drag, pressure drag and base drag. The drag of the tail is calculated similarly to the drag of the wing.

The velocity regions air-to-air missiles normally operate in result in turbulent flow along the skin. The methods used to calculate zero lift drag on both the body and the wing are referenced to the coefficient of friction of a flat plate with the equivalent characteristic length used to compute the Reynolds number. For the body the characteristic length is the length of the entire body and for the wing the mean aerodynamic chord (MAC) is the characteristic length. The incompressible flat plate skin friction coefficient from Lindsey and Redman (1980) is:

$$c_{fi} = \frac{0.455}{(\log 10(\mathrm{Re}))^{2.58}} \tag{5.4}$$

Where the Reynolds number is:

$$\mathrm{Re} = \frac{\rho V x}{\mu} \tag{5.5}$$

The incompressible skin friction coefficient is corrected for compressibility in the subsonic region by (Chin, 1961):

$$c_f = \frac{c_{fi}}{1 + 0.08\mathrm{M}^2} \tag{5.6}$$

and in the supersonic region by (Nicolai, 1984):

$$c_f = \frac{c_{fi}}{\left(1 + 0.144\mathrm{M}^2\right)^{0.65}} \tag{5.7}$$

Drag effects from wing-body interaction are assumed to be negligible. This is a valid assumption for wings mounted on bodies experiencing no expansion or contraction. Since the wing panels are mounted on a cylindrical body, the assumption is reasonable (Nielsen, 1960).

Chin (1961) suggests increasing the skin friction drag by 10% to help account for variations. The method used by MSLDSN is to increase the value of the coefficient of friction used in all drag calculations by 10%. The user has the option to change this value, called the friction factor, in all m-files used to calculate drag (e.g. CDOWBT).

Drag from protuberances, e.g., launch lugs, will also increase the zero lift drag of the missile. From Briggs (1991):

> Often the requirement to launch a given missile from several different aircraft using existing launchers leads to some rather esoteric constraints and arrangements of launch hooks and lugs, which produce important aerodynamic effects that are difficult to predict. For example, launch lugs, hooks, and other protuberances increase the zero-lift drag of air-to-air missiles by 20% on larger missiles to 100% on the smaller ones.

The user can simimulate this effect by altering the value of the friction factor or by increasing the final value of the zero-lift drag.

1. Body Drag

The base drag calculation requires a knowledge of the geometry of the nozzle of the propulsion system. The assumption is made that during propulsion system operation, base drag is zero. The base drag calculations techniques are included for power off operation.

a. Subsonic

In the subsonic region, the zero lift drag coefficient for the body from DATCOM (Hoak, 1978) is:

$$C_{D0} = C_{Df} + C_{DP} + C_{Db} \tag{5.8}$$

$$C_{Df} = c_f \frac{S_{WET}}{S_{ref}} \tag{5.9}$$

$$C_{DP} = c_f \left[\frac{60}{(l/d)^3} + 0.0025\left(\frac{l}{d}\right) \right] \frac{S_{WET}}{S_{ref}} \tag{5.10}$$

Where the wetted surface area of the body less the base is:

$$S_{WET} = (l - l_N)\pi d + S_{ogive} \tag{5.11}$$

$$S_{ogive} = \pi d^2 \left\{ \left[\left(\frac{l_N}{d}\right)^2 + \frac{1}{4} \right]^2 \sin^{-1} \left[\frac{\left(\frac{l_N}{d}\right)}{\left[\left(\frac{l_N}{d}\right)^2 + \frac{1}{4} \right]} \right] - \left(\frac{l_N}{d}\right) \left[\left(\frac{l_N}{d}\right)^2 - \frac{1}{4} \right] \right\} \tag{5.12}$$

The base drag coefficient is:

$$C_{Db} = 0.029 \frac{\left(\frac{d_b}{d}\right)^3}{\sqrt{C_{Df}}} \tag{5.13}$$

MSLDSN assumes the diameter of the base equals the diameter of the body, thus simplifying Equation (5.13).

At subsonic Mach numbers greater than 0.6, the skin friction and pressure drag are assumed to be constant and equal to the skin friction and pressure drag calculated at Mach 0.6. The base drag coefficient for Mach > 0.6 is calculated for values at Mach 0.6 and then corrected for Mach number using the method of DATCOM (Hoak et al, 1978).

b. Supersonic

In the supersonic region, the zero lift drag coefficient from DATCOM (Hoak et al, 1978) is:

$$C_{D0} = c_f \frac{S_{WET}}{S_{ref}} + C_{DP} + C_{Db} + C_W \tag{5.14}$$

Where the wetted surface area is the same as for the subsonic case. The wave drag of the nose is empirically derived by Miles (Chin, 1961) to be:

$$C_W = P\left\{1 - \frac{\left[196(l_N/d)^2 - 16\right]}{14(M+18)(l_N/d)^2}\right\} \tag{5.15}$$

$$P = \left(0.083 + \frac{0.096}{M^2}\right)\left(\frac{\sigma}{10}\right)^{1.69} \tag{5.16}$$

$$\sigma = 2\tan^{-1}\left[\frac{d}{2\,l_N}\right]\left(\frac{180^o}{\pi}\right) \tag{5.17}$$

The base drag coefficient for the supersonic case from DATCOM (Hoak, 1978) is:

$$C_{Db} = -C_{Pb}\left(\frac{d_b}{d}\right)^2 \tag{5.18}$$

Where the base pressure coefficient is calculated for a body with $d_b = d$. The pressure drag is assumed to linearly decrease from the value at Mach = 1 to zero at Mach = 1.2.

2. Wing drag

a. Subsonic

The zero lift drag for a wing with a double wedge profile operating in the subsonic region is (Krieger et al, 1991):

$$C_{D0} = c_f\left[1 + 1.2\left(\frac{t}{c}\right) + 60\left(\frac{t}{c}\right)^4\right]\frac{S_{WET}}{S_W} \tag{5.19}$$

The wetted surface area of the wing is twice the wing area.

$$S_{WET} = 2\,S_W \tag{5.20}$$

b. Supersonic

In the supersonic region the zero lift drag coefficient from DATCOM (Hoak et al, 1978) is:

$$C_{D0} = c_f \frac{S_{WET}}{S_W} + C_{DW} \tag{5.21}$$

The wave drag of the wing depends on the wing shape and whether the wing has a supersonic or subsonic leading edge. For double wedge profile wings with a supersonic leading edge:

$$C_{DW} = \frac{4}{\beta}\left(\frac{t}{c}\right)^2 \tag{5.22}$$

With subsonic leading edges:

$$C_{DW} = 4\left(\frac{t}{c}\right)^2 \cot(\Lambda_{LE}) \tag{5.23}$$

3. Transonic

Drag in the transonic region is difficult to calculate. The procedure used in MSLDSN is inspired by the method for calculating the wing-body-tail coefficient of drag in DATCOM (Hoak et al, 1978). The drag divergence Mach number is defined as the Mach number where the change in the drag coefficient with respect to Mach number equals 0.1.

$$\left(\frac{\partial C_D}{\partial M}\right)_D = 0.1 \tag{5.24}$$

In MSLDSN, the assumed drag divergence Mach number for the wing-body-tail combination is Mach 0.95. Mach 0.95 is selected based on the drag divergence Mach number of a slender body. For slender bodies with fineness ratios greater than six, the drag divergence Mach number is greater than or equal to Mach 0.95. From DATCOM (Hoak et al, 1978), the maximum drag coefficient of the wing-body-tail combination in the transonic region is the supersonic drag coefficient at Mach 1.1. Assuming the drag coefficient approximates a straight line from the drag divergence Mach number to the

supersonic drag coefficient at Mach 1.1, the drag coefficient values for Mach numbers in the transonic region becomes:

$$C_{D0}(M) = \left[\frac{C_{D0}(1.1) - C_{D0}(0.95)}{1.1 - 0.95}\right][M - 0.95] + C_{D0}(0.95) \qquad (5.25)$$

Figure 10 is a plot of the zero-lift drag coefficient versus Mach number in the transonic region for a missile. The linear approximation should be sufficient if the missile traverses the transonic region quickly. If the missile is being designed to operate in or near the transonic region, another technique is required and is beyond the capabilities of MSLDSN.

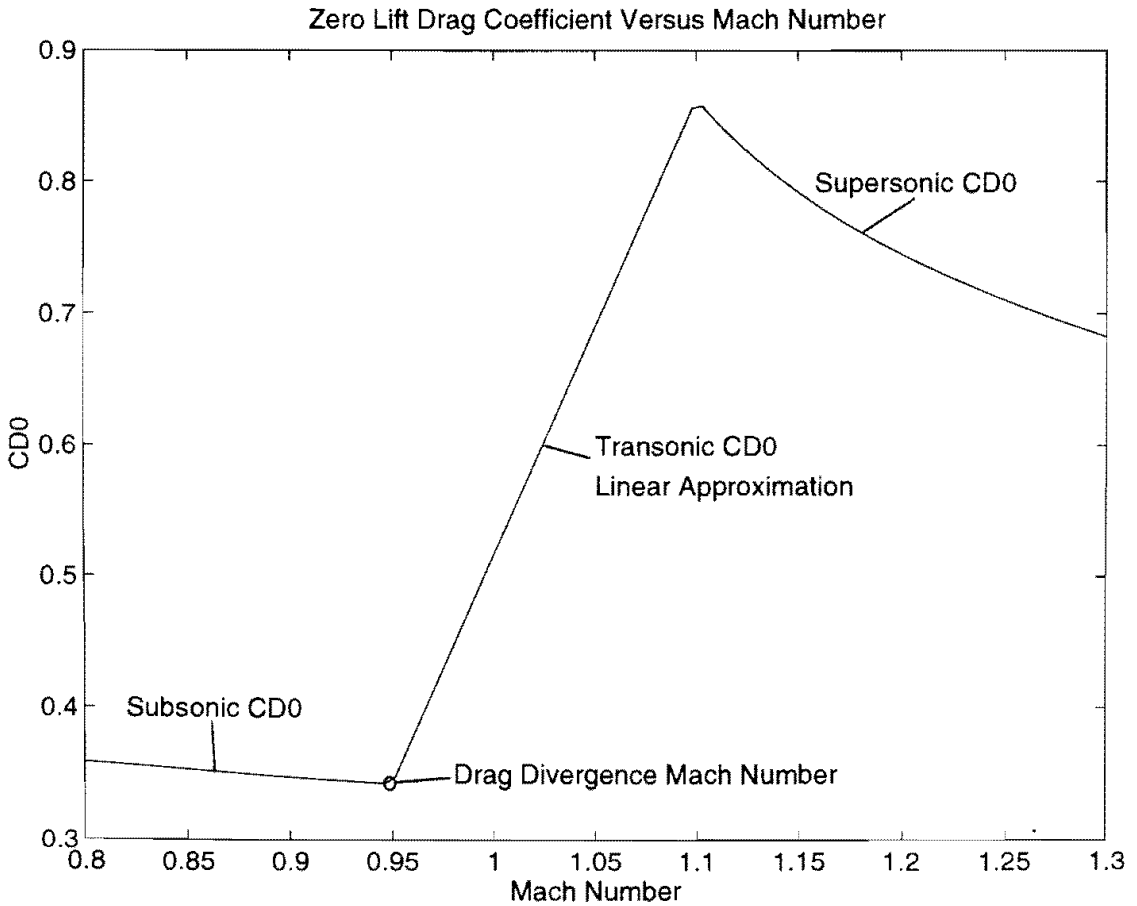


Figure 10. Linear Approximation of C_{D0} in the Transonic Region

4. Example One

CDOWBT calculates the coefficient of zero lift drag for a wing-body-tail combination. DRAGEX1 uses CDOWBT to calculate the C_{D0} for the missile designed at the end of Chapter III and plots the total drag versus Mach number (Figure 11).

B. DRAG AT TRIM

For the purposes of the following discussion, the trim condition is considered to exist when the missile is generating a steady-state lift force and the moment about the missile's center of gravity is zero. The equation for the drag of the missile at the trim

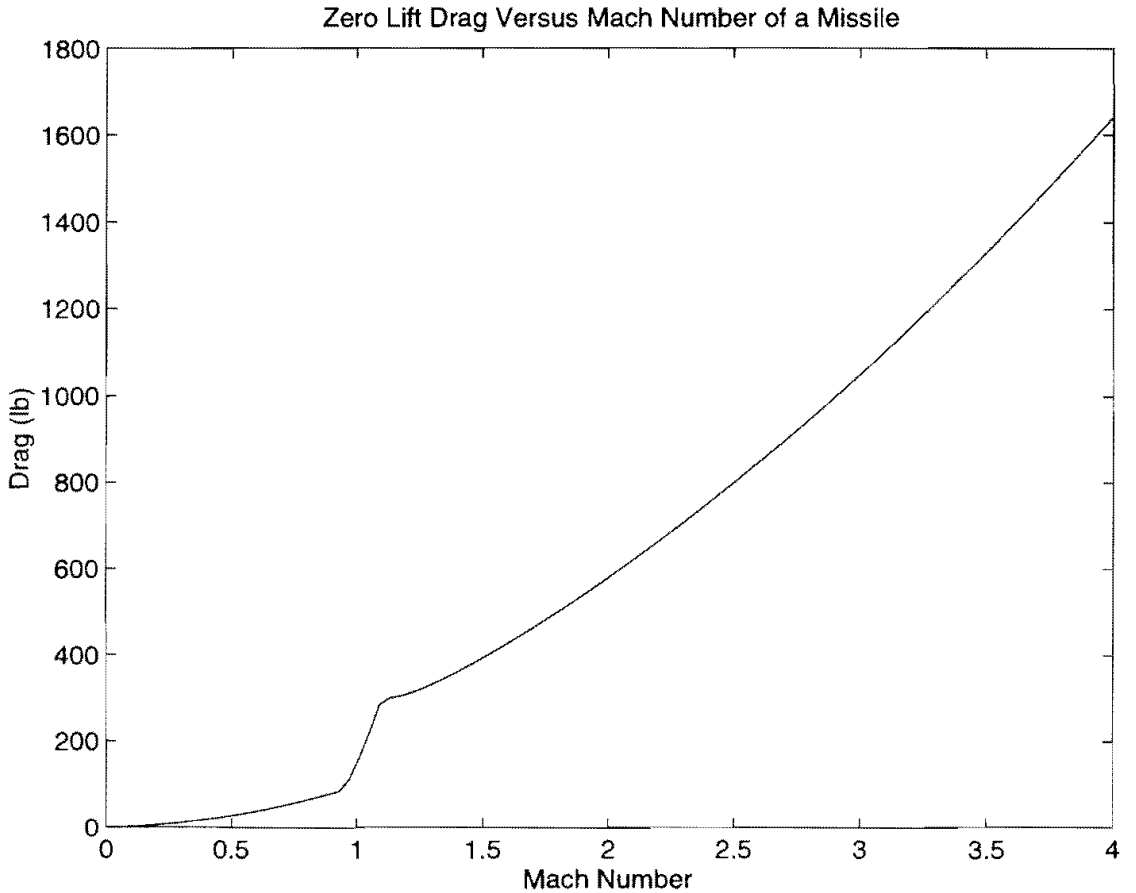


Figure 11. Drag Versus Mach Number

$$D = (C_{D0WB} + C_{D0V} + C_{DiWB} + C_{DTR})\bar{q}S_{ref} \tag{5.26}$$

For the cruciform missile geometry, the zero lift drag for the vertical surfaces will be included in the zero lift drag term of the wing-body combination.

$$C_{D0WB} + C_{D0V} = C_{D0WB} = C_{D0B} + 2C_{D0W}\frac{S_W}{S_{ref}} + C_{D0T}\frac{S_T}{S_{ref}} \tag{5.27}$$

The induced drag of the wing-body combination is:

$$C_{DiWB} = C_{DLW}\frac{S_W}{S_{ref}} + (C_{DB})_\alpha \tag{5.28}$$

The body drag coefficient due to angle of attack $[(C_{DB})_\alpha]$, the coefficient of drag due to lift of the wing and the change in drag due to trim are will be addressed individually.

1. Body Drag Due to Angle of Attack

The technique for calculating the coefficient of body drag due to angle of attack is derived from the component of the body normal force in the drag direction:

$$(C_{DB})_\alpha = C_{LB}\sin\alpha = C_{LB}\alpha = C_{LB\alpha}\alpha^2 \tag{5.29}$$

Thus the coefficient of drag due to angle of attack becomes:

$$(C_{DB})_\alpha = 2(k_2 - k_1)\alpha^2 + \eta c_{dc}\frac{A_p}{S_{ref}}\alpha^3 \tag{5.30}$$

The drag proportionality factor, η, equals one for supersonic flow and is a function of the fineness ratio of the body for subsonic flow. (Jerger, 1960)

2. Drag Due to Lift of the Wing

a. Subsonic

The coefficient of drag due to lift for wings with sharp leading edges and no leading edge suction (based on wing area) is inversely proportional to the lift-curve-slope (Krieger et al, 1991).

$$C_{DL} = \frac{C_L^2}{1.1C_{L\alpha}} = \frac{(C_{L\alpha}\alpha)^2}{1.1C_{L\alpha}} = \frac{C_{L\alpha}\alpha^2}{1.1} \tag{5.31}$$

b. Supersonic

For wings with supersonic leading edges, the drag due to lift is based on linear theory (Krieger et al, 1991).

$$C_{DL} = \frac{\beta C_L^2}{4} = \frac{4\alpha^2}{\beta} \tag{5.32}$$

For wings with subsonic leading edges the procedure from DATCOM (Hoak et al, 1978) for supersonic wings in the parabolic-drag region is used.

$$C_{DL} = C_L^2 \left[\pi A \frac{C_{DL}}{C_L^2} \frac{p}{1+p} \right] \left(\frac{1}{\pi A} \right) \left(\frac{1+p}{p} \right) \tag{5.33}$$

$$p = \frac{S_W}{b_W \ell} \tag{5.34}$$

The term $\left[\pi A \frac{C_{DL}}{C_L^2} \frac{1+p}{p} \right]$ is an empirical relationship which is a function of $\frac{\beta b_W}{2\ell}$. For a triangular planform ℓ is c_{rW} and thus p simplifies to:

$$p = \frac{S_W}{b_W \ell} = \frac{\frac{1}{2} b_W c_{rW}}{b_W c_{rW}} = \frac{1}{2} \tag{5.35}$$

$$\left(\frac{1+p}{p} \right) = \frac{1+\frac{1}{2}}{\frac{1}{2}} = \frac{\frac{3}{2}}{\frac{1}{2}} = 3 \tag{5.36}$$

Thus the coefficient of drag due to lift becomes:

$$C_{DL} = 3(C_{L\alpha}\alpha)^2 \left[\pi A \frac{C_{DL}}{C_L^2} \frac{1+p}{p} \right] \left(\frac{1}{\pi A} \right) \tag{5.37}$$

Empirical data of wings with round and sharp leading edges was used in deriving $\left[\pi A \frac{C_{DL}}{C_L^2} \frac{1+p}{p} \right]$. The range of wing parameters with sharp leading edges used to calculate $\left[\pi A \frac{C_{DL}}{C_L^2} \frac{1+p}{p} \right]$ is:

$$1.5 \leq A \leq 3.5$$

$$0 \le \Lambda_{LE} \le 71^\circ$$

$$0 \le \lambda \le 1$$

$$0.333 \le p \le 0.995$$

The technique will be used even if the wing parameters are outside the range of the empirical data. Most missile planforms encountered will fall within the specified range.

3. Drag Due to Trim

Drag due to trim is drag as a result of the tail deflection required to maintain the trim angle of attack. The procedure used herein loosely follows the procedure in DATCOM (Hoak et al, 1978). Drag due to trim is defined in the following manner in MSLDSN:

$$C_{DTR} = (C_{DT} \cos\delta_{TR} + C_{LT} \sin\delta_{TR}) \frac{S_T}{S_{ref}} \frac{\bar{q}_T}{\bar{q}} \tag{5.38}$$

Where:

$$C_{DT} = C_{D0T} + C_{DLT} \tag{5.39}$$

C_{DLT} is calculated in the same manner as C_{DLW} using trim tail deflection angle vice angle of attack.

4. Trim Angle of Attack and Deflection Angle

In a trim condition, the lift generated by the missile is equal to the load factor times the missile's weight and the sum of the moments about the missile's center of gravity is zero.

$$(C_{L\alpha}\alpha_{TR} + C_{L\delta}\delta_{TR})\bar{q}S_{ref} = nW \tag{5.40}$$

$$C_{m\alpha}\alpha_{TR} + C_{m\delta}\delta_{TR} = 0 \tag{5.41}$$

Rearranging the terms and placing them in a matrix format:

$$\begin{bmatrix} C_{m\alpha} & C_{m\delta} \\ C_{L\alpha} & C_{L\delta} \end{bmatrix} \begin{bmatrix} \alpha_{TR} \\ \delta_{TR} \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{nW}{\bar{q}S_{ref}} \end{bmatrix} \tag{5.42}$$

The system of equations is easily solved in MATLAB to attain values of the trim angle of attack and trim deflection angle (Little et al, 1993). For example, let the variables *A, X and Y* represent:

$$A = \begin{bmatrix} C_{m\alpha} & C_{m\delta} \\ C_{L\alpha} & C_{L\delta} \end{bmatrix} \tag{5.43}$$

$$X = \begin{bmatrix} \alpha_{TR} \\ \delta_{TR} \end{bmatrix} \tag{5.44}$$

$$Y = \begin{bmatrix} 0 \\ \dfrac{nW}{\overline{q}S_{ref}} \end{bmatrix} \tag{5.45}$$

Then, the solution of:

$$A\,X = Y \tag{5.46}$$

for unknown X is:

$$X = A \setminus Y \tag{5.47}$$

5. Example Two

CDTRIMEX calculates the coefficient of drag for the missile from example one assuming the missile is trimmed in level flight (i.e. load factor = 1) and at a load factor of 10. CDTRIMEX then calculates the drag coefficient using only CDOWBT. The values accounting for lift drag are compared to the zero lift drag and the largest errors between the two is returned. The results are plotted in Figure 12.

```
» cdtrimex
maxer1g =
   0.0085
maxer10g =
   1.4676
```

Thus, the maximum error between the two calculations is less than 1% for load factor = 1 and almost 150% for load factor = 10. Considering the additional calculation complexity and the relatively small gain in accuracy, the zero-lift drag calculation should be of sufficient accuracy in the case of low load factors. The additional accuracy of the drag due to lift would become important when the missile is experiencing large lift forces. In these conditions, the added complexity of the calculations would be justified.

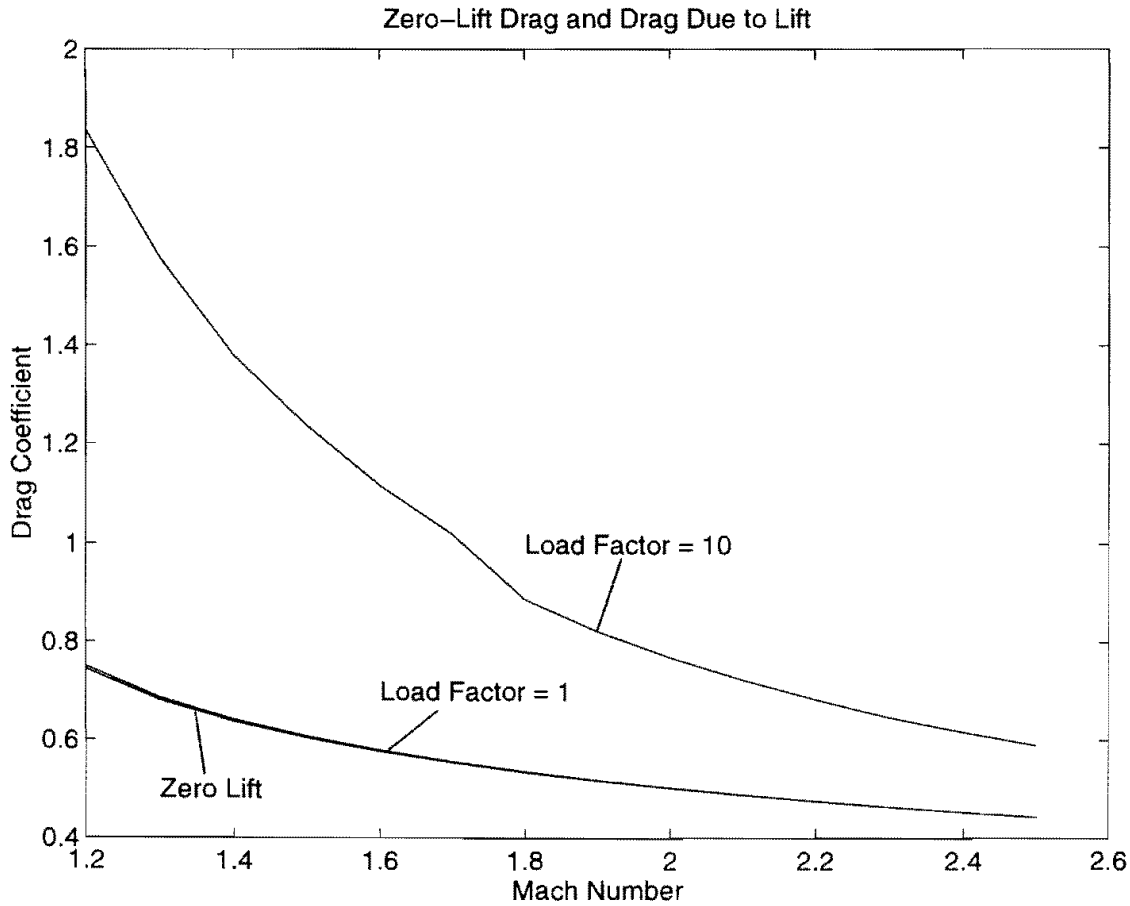


Figure 12. Comparison of Zero Lift Drag and Drag Due to Lift

VI. MISSILE PERFORMANCE AND PROPULSION

Missile performance characteristics, i.e. maximum velocity and range, can be estimated once the forces on the missiles are known. The derivation of the characteristics of the propulsion system, i.e. weight and length, are intimately related with missile performance and will be treated together.

Two major types of thrust scheduling are used by rocket-powered air-to-air missiles resulting in two different trajectories; boost-glide and boost-sustain. In the boost-glide trajectory, the rocket motor burns at maximum thrust for a short time period (2 - 5 sec), accelerating the missile to a high velocity (Mach = 4 - 6). The rest of the missile's flight is unpowered and the missile slowly decelerates. In boost-sustain, the missile is rapidly accelerated to a cruise velocity (Mach = 2 - 3.5) by the booster. The sustainer then provides sufficient thrust to overcome missile drag at cruise. The boost-glide trajectory is used in short and medium range missiles. The boost-sustain trajectory is used by medium and long range missiles. An analysis of boost-glide trajectories is also helpful in analyzing the boost phase of a boost-sustain missile. For very long range missiles, the rocket motor can be designed to provide multiple thrust pulses. The resulting trajectory is a series of boost-glide trajectories. (Netzer, 1993)

A. INCREMENTAL VELOCITY DUE TO BOOST

The velocity of the missile after booster burnout minus the velocity of the missile prior to boost is defined as the incremental velocity due to boost, ΔV_b. The derivation of the ΔV_b can be found in Chin (1961), Jerger(1960) and Lindsey and Redman (1980). The nomenclature of Lindsey and Redman was the most consistent and is used in the following derivation of ΔV_b.

The incremental velocity due to boost is derived from the axial forces acting on a missile. Assuming the angle of attack of the missile is small (Figure 13):

$$m\frac{dV}{dt} = \mathrm{T} - \mathrm{D} - \mathrm{W}\sin(\gamma) \tag{6.1}$$

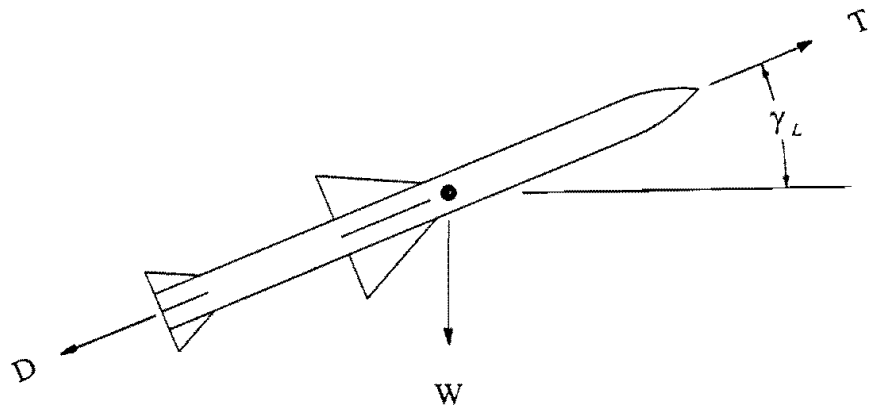


Figure 13. Axial Forces

$$m = \frac{W}{g} \tag{6.2}$$

Rearranging terms and substituting mass with weight:

$$dV = g\left(\frac{T-D}{W}\right)dt - g\sin(\gamma)dt \tag{6.3}$$

Integrating both sides of the equation:

$$\int_{V_L}^{V_f} dV = \int_0^{t_b} g\left(\frac{T-D}{W}\right)dt - \int_0^{t_b} g\sin(\gamma) \tag{6.4}$$

Assuming a constant thrust and that the weight of the missile varies linearly with time during boost, the weight of the missile at any given time during boost is:

$$W(t) = W_L - W_P\left(\frac{t}{t_b}\right) \tag{6.5}$$

Where the weight of the propellant is equal to the launch weight of the missile minus the weight of the missile at the end of burn.

$$W_P = W_L - W_E \tag{6.6}$$

In the first approximation, if the drag is assumed to be a constant value, the acceleration of the missile is constant. (Later in this chapter, the actual drag value will be used.) If the flight path angle does not vary from launch and completing the integration, the incremental velocity due to boost is:

$$\Delta V_b = V_f - V_L = g\int_0^{t_b} \frac{(T-D)t_b}{W_L t_b - W_P t} - g t_b \sin(\gamma_L) \tag{6.7}$$

$$\Delta V_b = g\frac{(T-D)t_b}{W_P}\ln\left[\frac{(W_L - W_P)t_b}{W_L t_b}\right] - g t_b \sin(\gamma_L) \tag{6.8}$$

$$\Delta V_b = g\frac{(T-D)t_b}{W_P}\ln\left[\frac{W_L}{W_E}\right] - g t_b \sin(\gamma_L) \tag{6.9}$$

The specific impulse of a rocket motor can be represented by:

$$I_{sp} = \frac{T t_b}{W_P} \tag{6.10}$$

Resulting in:

$$\Delta V_b = g\left(1 - \frac{D}{T}\right) I_{sp} \ln\left[\frac{W_L}{W_E}\right] - g t_b \sin(\gamma_L) \tag{6.11}$$

B. BOOSTER WEIGHT AND LENGTH

1. Booster Weight

Equation (6.11) can be manipulated to provide an initial weight and length estimate of the propulsion subsection of the booster. As a first approximation, the drag-to-thrust ratio of the missile during boost is assumed to be small ($\frac{D}{T} << 1$). If the flight path angle is also assumed to be zero, the weight of the propellant becomes:

$$\ln\left[\frac{W_E}{W_L}\right] = \frac{-\Delta V_b}{g I_{sp}} \tag{6.12}$$

$$\frac{W_E}{W_L} = e^{\frac{-\Delta V_b}{g I_{sp}}} \tag{6.13}$$

$$W_P = W_L\left(1 - e^{\frac{-\Delta V_b}{g I_{sp}}}\right) \tag{6.14}$$

The ratio between the weight of the propellant and the weight of the complete rocket motor is the rocket motor mass ratio.

$$\zeta_m = \frac{W_P}{W_m} \tag{6.15}$$

The value of ζ_m varies from 0.75 for small rocket motors operating at large chamber pressures to 0.9 for larger motors operating at reduced pressures (Chin, 1961). Thus the weight of the booster motor is:

$$W_m = \frac{W_P}{\zeta_m} = \left(\frac{W_L}{\zeta_m}\right)\left(1 - e^{\frac{-\Delta V_b}{g I_{sp}}}\right) \tag{6.16}$$

If an estimate of drag is available, a value for the average drag can be used. The weight of the booster motor then becomes:

$$\ln\left[\frac{W_E}{W_L}\right] = \frac{-\Delta V_b}{g\left(1 - \frac{\overline{D}}{T}\right) I_{sp}} \tag{6.17}$$

$$\frac{W_E}{W_L} = e^{\frac{-\Delta V_b}{g\left(1 - \frac{\overline{D}}{T}\right) I_{sp}}} \tag{6.18}$$

$$W_m = \left(\frac{W_L}{\zeta_m}\right)\left[1 - e^{\frac{-\Delta V_b}{g\left(1 - \frac{\overline{D}}{T}\right) I_{sp}}}\right] \tag{6.19}$$

2. Booster Length

The two major components in the length of the booster is the length of the propellant and the length of the nozzle.

$$\text{PLEN} = l_{nz} + l_P \tag{6.20}$$

The length of the propellant assuming an end-burning grain is:

$$l_{eb} = \frac{4W_P}{\pi d^2 \rho_P} \tag{6.21}$$

For internal-burning motors, e.g., star grains, the propellant does not fill the entire motor casing. The ratio of the propellant volume to the volume of the casing is the volumetric packing factor. A reasonable value for the volumetric packing factor is 0.85 (Chin, 1961). Therefore, the length of the propellant for an internal-burning motor is:

$$l_{ib} = \frac{l_{eb}}{v_P} = \frac{4W_P}{v_P \pi d^2 \rho_P} \tag{6.22}$$

Calculating the nozzle length analytically requires significant knowledge of the propellant and casing properties. Chin (1961) recommends the length of the nozzle be estimated as 30-35% of the length of the propellant. Assuming a value of 30%, the length of the booster is:

$$\text{PLEN} = 1.3\, l_P = 1.3 \frac{4W_P}{v_P \pi d^2 \rho_P} \tag{6.23}$$

C. SUSTAINER WEIGHT AND LENGTH

The thrust produced by the sustainer equals the drag of the missile at the cruise velocity. The burn time of the sustainer must be of sufficient duration for the missile to reach its maximum range.

$$t_s = \frac{R_{max} - D_b}{V_C} \tag{6.24}$$

Therefore, the weight of the sustainer propellant is:

$$W_P = \frac{Dt_s}{I_{sp}} \tag{6.25}$$

The weight and length of the sustainer can then be calculated using the same procedure outlined for the booster.

Since the assumption was made that the missile underwent constant acceleration during the boost phase, the distance traveled during boost is obtained by integrating Equation (6.9) and correcting for the flight path angle:

$$D_b = \left[g\left\{ \frac{(T-D)}{W_P} \ln\left[\frac{W_L}{W_E} \right] - \sin(\gamma_L) \right\} \frac{t_b^2}{2} + V_L t_b \right] \cos(\gamma_L) \tag{6.26}$$

$$t_b = \frac{I_{sp} W_P}{T} \tag{6.27}$$

The same drag assumption used in estimating propellant weight should be used when estimating distance traveled during boost.

D. PROPELLANT PROPERTIES

Representative values of propellant properties are required to solve for values of propulsion subsection weight and length. Table 4 from Lindsey and Redman (1980) provides a range of values for propellant specific impulse and density.

	Sustainer	Booster
I_{sp}	180-210	210-260
ρ_P	102-107.1	107.1-112.3

Table 4. Representative Propellant Properties

E. EXAMPLE ONE

PROPEX1 calculates the propulsion length and weight for a missile assuming the drag is zero at boost and then for the average drag.

```
» propex1
Assume: boost to Mach =4, drag = 0, Isp = 260 sec and rho = 104 lb/ft^3
```

pwt =

151.6461

plen =

7.0986

Assume: drag = average drag, T = 10000 and other values the same.

pwt =

159.4989

plen =

7.4662

F. NUMERICAL INTEGRATION OF ΔV_b

When Equation (6.4) was initially solved, assumptions were made for drag to ease the integration. Equation (6.4), without the flight path term, can be written in the following form:

$$\frac{dV}{dt} = \frac{gT(t)}{W(t)} - \frac{\rho V^2 S_{ref} C_{D0}(V)}{2W(t)} \tag{6.28}$$

Equation (6.28) can be solved numerically to provide a better estimate of missile performance.

One group of numerical integration techniques is known as multistep methods. Multistep methods take advantage of previous calculations of the function and its derivative. Multistep methods typically use equally spaced intervals to simply the calculations. Since information from previous calculations is necessary to use multistep methods, a single-step method is usually required to establish the values of the first few steps of the function. Numerical integration techniques can be found in Gerald and Wheatley (1994) and in Chin (1961).

An example of a second degree multistep method for solving Equation (6.28) for velocity and distance is:

$$V_n = V_{n-1} + (3a_{n-1} - a_{n-2})\frac{\Delta t}{2} \tag{6.29}$$

$$D_n = D_{n-1} + (3V_{n-1} - V_{n-2})\frac{\Delta t}{2} \tag{6.30}$$

Where $a = \frac{dV}{dt}$. The values for the first two steps of the integration are calculated using:

$$V_1 = a_1 \Delta t + V_L \tag{6.31}$$

$$D_1 = V_1 \Delta t \tag{6.32}$$

$$V_2 = V_1 + (a_1 + a_2)\frac{\Delta t}{2} \tag{6.33}$$

$$D_2 = D_1 + (3V_1 - V_L)\frac{\Delta t}{2} \tag{6.34}$$

PROPEX2 uses data for the missile designed at the end of Chapter III and plots the boost-glide trajectory of the missile by iterating Equation (6.28) using the previous method. The thrust of the missile is assumed to be constant during boost. Figures 14 and 15 are the resulting velocity profile and range plot. The algorithm in PROPEX2 is easily copied for use by the user in script m-files. The user should replace the missile parameters in PROPEX2 with their own and save the file under another name.

A more powerful and more computer intensive method involves the use of a third degree multistep method and employing a Runge-Kutta single-step method for evaluating the initial points. The steps for solving missile velocity are presented. First, the first two increments of velocity are solved for using a fourth-order Runga-Kutta method.

$$V_n = V_L + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{6.35}$$

Where:

$$k_1 = a(t_n, V_n)\Delta t \tag{6.36}$$

$$k_2 = a\left(t_n + \frac{\Delta t}{2}, V_n + \frac{k_1}{2}\right)\Delta t \tag{6.37}$$

$$k_3 = a\left(t_n + \frac{\Delta t}{2}, V_n + \frac{k_2}{2}\right)\Delta t \tag{6.38}$$

$$k_4 = a(t_n + \Delta t, V_n + k_3)\Delta t \tag{6.39}$$

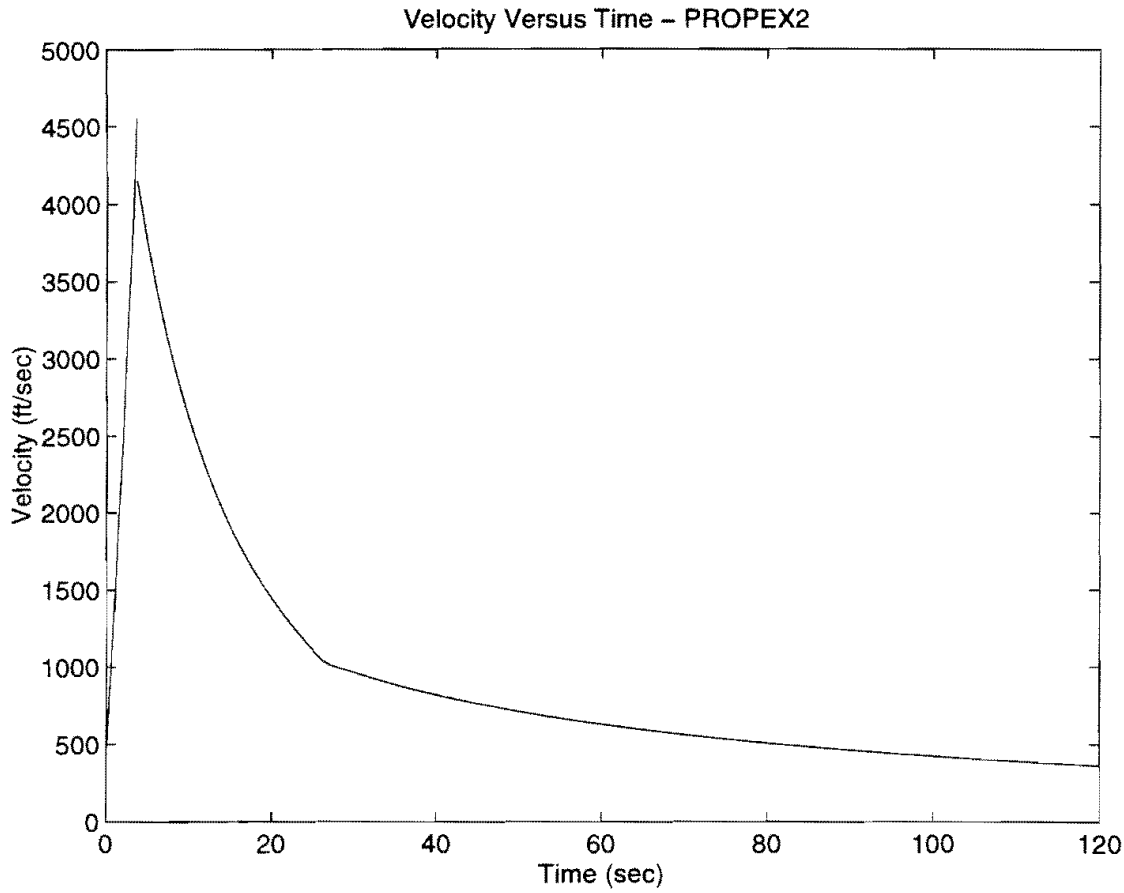


Figure 14. PROPEX2 Velocity Profile

Then the second degree multistep method is used to establish the third increment:

$$V_n = V_{n-1} + (3a_{n-1} - a_{n-2})\frac{\Delta t}{2} \tag{6.40}$$

The fourth and subsequent increments are solved for by using a third degree multistep method:

$$V_n = V_{n-1} + \frac{\Delta t}{12}(23a_{n-1} - 16a_{n-2} + 5a_{n-3}) \tag{6.41}$$

Figures 16 and 17 result.

The two methods compare closely in velocity and range. There are more powerful numerical integration methods available, but considering the relative inaccuracy in the

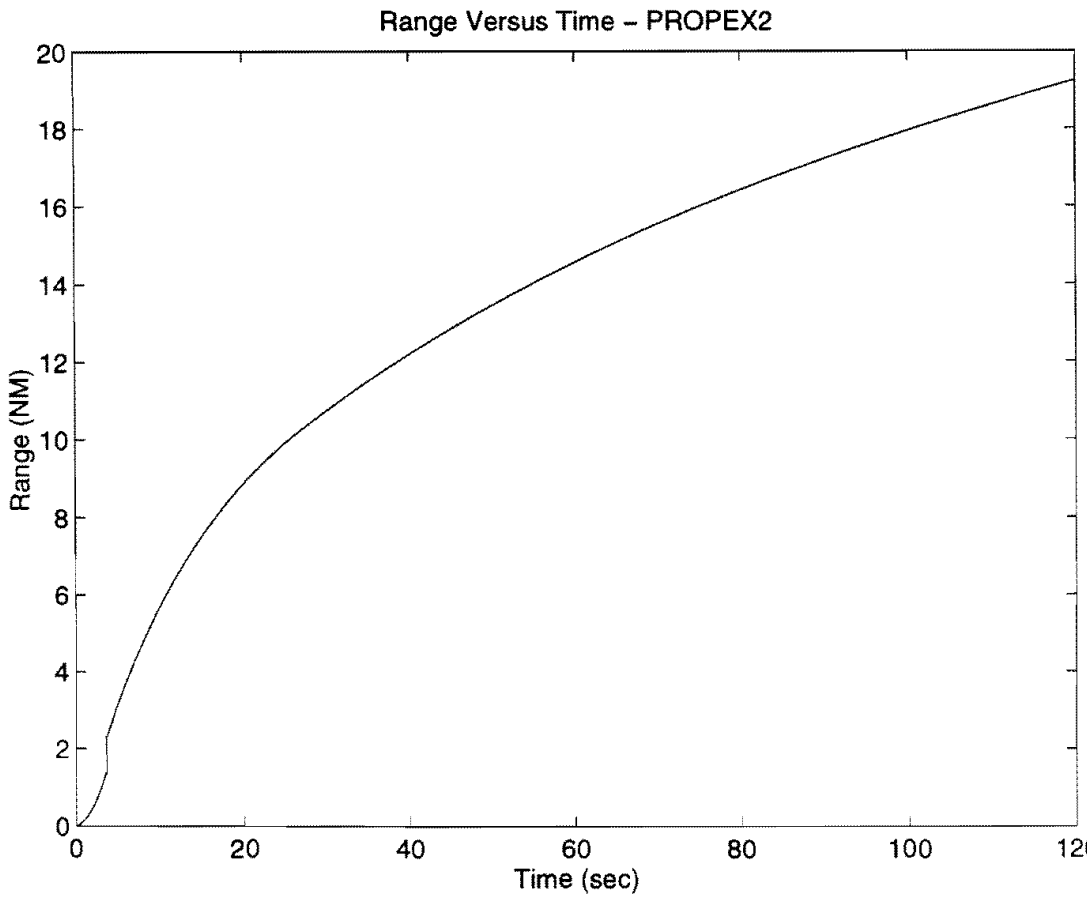


Figure 15. PROPEX2 Range Plot

knowledge of the actual drag coefficient, the increased complexity would prove of little value. If the range plot has a discontinuity in its shape or if MATLAB returns a warning:

WARNING: Complex parts of X and/or Y arguments ignored.

The user should consider increasing the number of iterations from the default value of 100.

PROPEX4 is an example of a boost-glide-boost-glide trajectory. The m-file can handle more complex trajectories such as boost-sustain or multi-pulse trajectories.

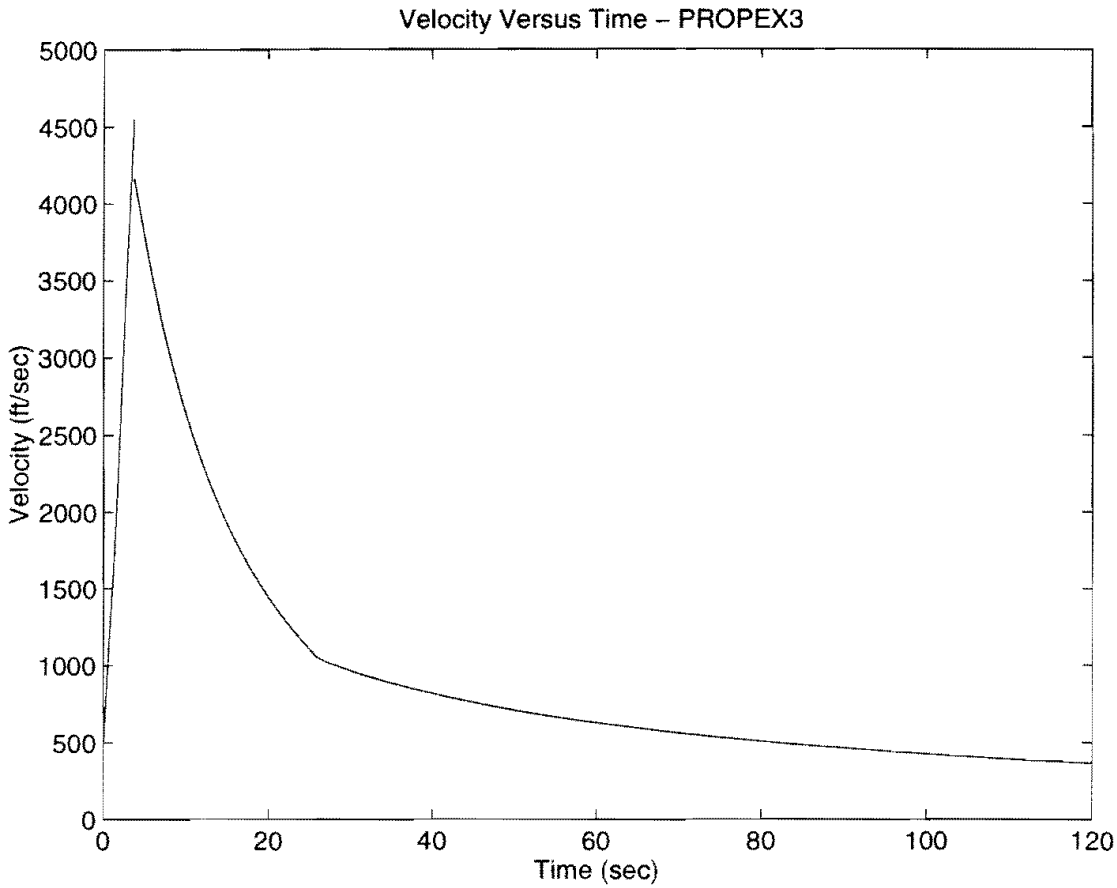


Figure 16. PROPEX3 Velocity Profile

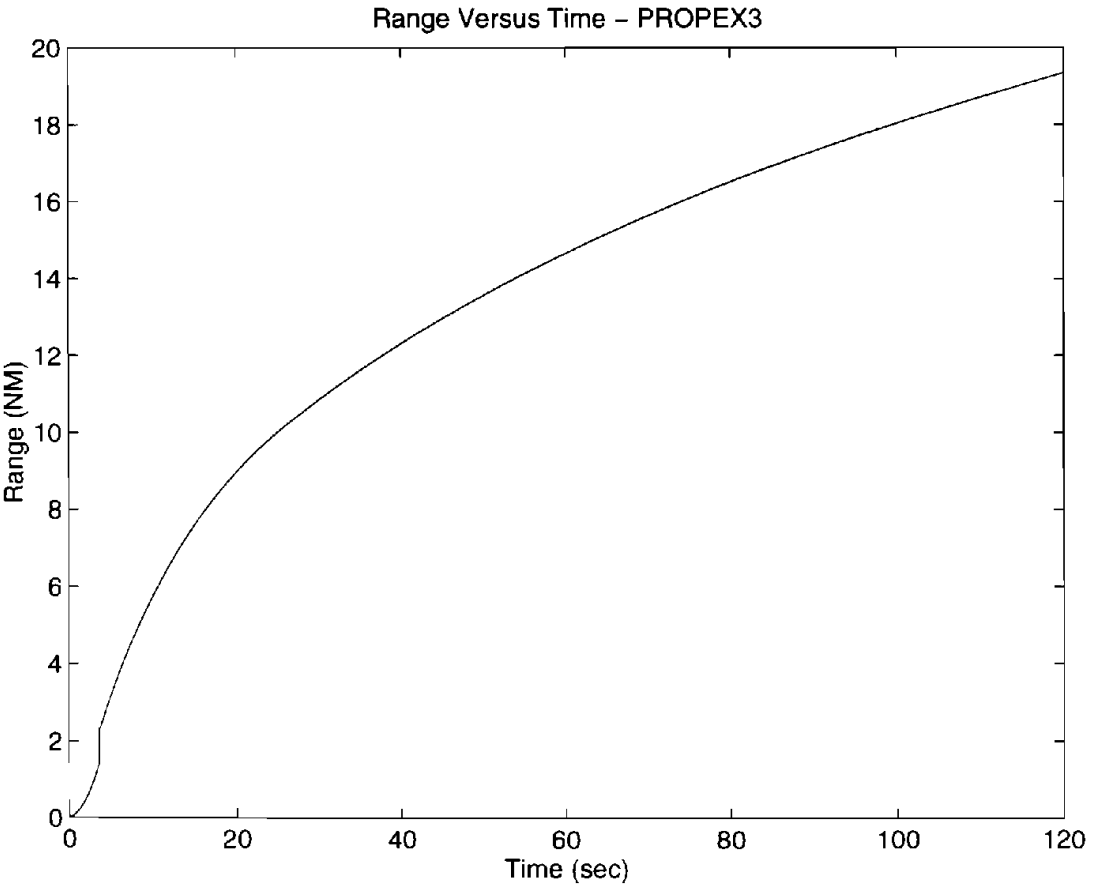


Figure 17. PROPEX3 Range Plot

VII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The Missile Design Toolbox provides the missile designer with a fast and easy-to-use collection of computer-based tools to aid in establishing and evaluating missile parameters during the conceptual phase of design. Developed using The MathWorks' MATLAB interactive computing environment, MSLDSN takes advantage of MATLAB's powerful numerical and graphical capabilities. By addressing more than one design discipline, e.g, aerodynamics, under a common computing environment, the designer can easily manipulate data to conduct cross-discipline analysis.

B. RECOMMENDATIONS

MSLDSN provides a core of MATLAB m-files which can be expanded to include more disciplines of missile design, other missile types or a more in-depth analysis of a design discipline already covered. Some specific examples of expansion might include:

- Guidance system performance
- Surface-to-air missiles
- Delta planform wings

Each new addition to MSLDSN should be treated as a separate module by the developer. The user makes the decision on what information to use from each module. For example, the user can decide to either use the missile subsection weights developed in Chapter II of this thesis or weights based on other sources. Thus, the user can use the weights from Chapter II as an initial value and then, later, use a better estimate derived from a more specific or detailed analysis.

APPENDIX A. LIFT COEFFICIENTS

Lift coefficients are calculated using techniques from NACA Report 1307 (Pitts et al, 1953). The following equations and discussion are valid for the tail as well as the wing.

1. Lift on the wing in presence of the body for variable angle of attack is calculated from slender-body theory and is valid for slender wing-body combinations. See BIGKWB.

$$K_{W(B)} = \frac{2}{\pi} \frac{\left\{\left(1+\frac{r^4}{s^4}\right)\left[\frac{1}{2}\tan^{-1}\left(\frac{s}{r}-\frac{r}{s}\right)+\frac{\pi}{4}\right]-\frac{r^2}{s^2}\left[\left(\frac{s}{r}-\frac{r}{s}\right)+2\tan^{-1}\left(\frac{r}{s}\right)\right]\right\}}{\left(1-\frac{r}{s}\right)^2} \tag{A.1}$$

2. Lift on the wing in presence of the body at variable wing incidence angle for triangular wing-body combinations is calculated from slender-body theory. See SMLKWB.

$$\tau = \frac{s}{r} \tag{A.2}$$

$$k_{W(B)} = \frac{1}{\pi^2}\left[\begin{array}{l}\frac{\pi(\tau^2+1)^2}{\tau^2(\tau-1)^2}\sin^{-1}\left(\frac{\tau^2-1}{\tau^2+1}\right)-\frac{2\pi(\tau+1)}{\tau(\tau-1)}+\frac{8}{(\tau-1)^2}\log\left(\frac{\tau^2+1}{2\tau}\right)+ \\ \frac{\pi^2}{4}\frac{(\tau+1)^2}{\tau^2}-\frac{4(\tau+1)}{\tau(\tau-1)}\sin^{-1}\left(\frac{\tau^2-1}{\tau^2+1}\right)+\frac{(\tau^2+1)^2}{\tau^2(\tau-1)^2}\left(\sin^{-1}\left(\frac{\tau^2-1}{\tau^2+1}\right)\right)^2\end{array}\right] \tag{A.3}$$

3. Lift on the body in the presence of the wing at variable wing incidence angle for cylindrical bodies comes from slender-body theory. See SMLKBW.

$$k_{B(W)} = K_{W(B)} - k_{W(B)} \tag{A.4}$$

4. Lift on the body in the presence of the wing depends on the presence of an afterbody, subsonic or supersonic leading edges and β. See BIGKBW for a summary.

Slender-body theory is valid for the supersonic wing condition $\beta A(1+\lambda)\left(\frac{1}{m\beta}+1\right)\leq 4$ and subsonic speeds. See BIGKBWSB.

$$K_{B(W)}=\frac{\left(1-\frac{r^2}{s^2}\right)^2-\frac{2}{\pi}\left\{\begin{array}{l}\left(1+\frac{r^4}{s^4}\right)\left[\frac{1}{2}\tan^{-1}\left(\frac{1}{2}\left(\frac{s}{r}-\frac{r}{s}\right)\right)+\frac{\pi}{4}\right]\\ -\frac{r^2}{s^2}\left[\left(\frac{s}{r}-\frac{r}{s}\right)+2\tan^{-1}\left(\frac{r}{s}\right)\right]\end{array}\right\}}{\left(1-\frac{r}{s}\right)^2} \qquad \text{(A.5)}$$

For a triangular wing, $\lambda=0$ and $m=\frac{A}{4}$, thus $\beta A(1+\lambda)\left(\frac{1}{m\beta}+1\right)\leq 4$ results in the condition that $\beta A\leq 0$. Thus slender body theory is not valid for a triangular planform wing in supersonic flow.

For the $\beta A(1+\lambda)\left(\frac{1}{m\beta}+1\right)>4$ the theory presented in NACA 1307 (Pitts et al, 1953) is used. The following equations are derived from the theory and simplified for triangular wing geometry.

For triangular wing-body combinations with an afterbody extending aft of the wing trailing edge and subsonic leading edges, $K_{B(W)}$ is: (See BKBWSBA)

$$K_{B(W)} = \frac{8\,E\left(\sqrt{1-\left(\frac{\beta A}{4}\right)^2}\right)}{\pi^2\left(\frac{\beta A}{4}\right)^2}\left\{\frac{\left(\frac{\beta A}{4}\right)^2}{2\left(\frac{\beta A}{4}+1\right)^2}\left[1+\frac{2\left(\frac{\beta A}{4}+1\right)\frac{r}{s}}{1-\frac{r}{s}}\right]^{3/2}-\left(\frac{\beta A}{\beta A+4}\right)^2+\right.$$

$$\left.\frac{\left(\frac{\beta A}{4}\right)^2}{2\left(\frac{\beta A}{4}+1\right)^2}\left[1+\frac{2\left(\frac{\beta A}{4}+1\right)\frac{r}{s}}{1-\frac{r}{s}}\right]^{1/2}-\right.$$

$$\left.2\left(\frac{\beta A}{4}\right)^2\left(\frac{\frac{r}{s}}{1-\frac{r}{s}}\right)^2\tanh^{-1}\sqrt{\frac{1}{1+2\frac{\left(\frac{\beta A}{4}+1\right)\frac{r}{s}}{\left(1-\frac{r}{s}\right)}}}\right\}$$

Equation (A.6)

For triangular wing-body combinations with an afterbody extending aft of the wing trailing edge and supersonic leading edges, $K_{B(W)}$ is: (See BKBWSPA)

$$K_{B(W)} = \frac{1}{\pi\sqrt{\left(\frac{\beta A}{4}\right)^2 - 1}} \left\{ \left(\frac{\beta A}{\beta A + 4}\right)\left[1 + \frac{2\left(1+\frac{\beta A}{4}\right)\frac{r}{s}}{1-\frac{r}{s}}\right]^2 \cos^{-1}\left[\frac{1+\frac{\frac{\beta A}{2}\left(1+\frac{\beta A}{4}\right)\frac{r}{s}}{1-\frac{r}{s}}}{\frac{\beta A}{4}+\frac{\frac{\beta A}{2}\left(1+\frac{\beta A}{4}\right)\frac{r}{s}}{1-\frac{r}{s}}}\right] + \right.$$

$$\frac{\sqrt{\left(\frac{\beta A}{4}\right)^2 - 1}}{\left(1+\frac{\beta A}{4}\right)}\sqrt{1+\frac{\beta A\frac{r}{s}}{1-\frac{r}{s}}} - \frac{\frac{\beta A}{4}}{1+\frac{\beta A}{4}}\cos^{-1}\left(\frac{4}{\beta A}\right) - \frac{\sqrt{\left(\frac{\beta A}{4}\right)^2 - 1}}{\left(1+\frac{\beta A}{4}\right)}$$

$$\left. -\beta A\sqrt{\left(\frac{\beta A}{4}\right)^2 - 1}\frac{\left(\frac{r}{s}\right)^2}{\left(1-\frac{r}{s}\right)^2}\cosh^{-1}\left[1+\frac{2\left(1-\frac{r}{s}\right)}{\beta A\frac{r}{s}}\right] \right\}$$

Equation (A.7)

For triangular wing-body combinations with no afterbody extending aft of the wing trailing edge, subsonic leading edges and $\beta < \frac{2\left(1-\frac{r}{s}\right)}{\frac{r}{s}A}$, $K_{B(W)}$ is: (See BKBWSBNA)

$$K_{B(W)} = \frac{16\sqrt{\frac{\beta A}{4}}\left(\frac{r}{s}\right)^2 E\left(\sqrt{1-\left(\frac{\beta A}{4}\right)^2}\right)}{\pi^2\left(\frac{\beta A}{4}+1\right)\left(1-\frac{r}{s}\right)^2}\left\{\left(\frac{\frac{r}{s}+1}{2\frac{r}{s}}\right)\sqrt{\left(\frac{2\left(1-\frac{r}{s}\right)}{\beta A\frac{r}{s}}-1\right)\left(\frac{\frac{r}{s}+1}{2\frac{r}{s}}\right)}-\left[\frac{2\left(1-\frac{r}{s}\right)}{\beta A\frac{r}{s}}\right]^2\left(\frac{\beta A}{4}\right)^{3/2}\right.$$

$$+\frac{\left(1-\frac{r}{s}\right)^2}{\beta A\left(\frac{r}{s}\right)^2}\left(\frac{\beta A}{4}+1\right)\left[\tan^{-1}\left(\sqrt{\frac{4}{\beta A}}\right)-\sqrt{\frac{\left(\frac{2\left(1-\frac{r}{s}\right)}{\beta A\frac{r}{s}}-1\right)}{\left(\frac{\frac{r}{s}+1}{2\frac{r}{s}}\right)}}\right]$$

$$\left.-\frac{\left(\frac{\beta A}{4}+1\right)}{\sqrt{\frac{\beta A}{4}}}\tanh^{-1}\sqrt{\left(\frac{\beta A}{4}\right)\frac{\left(\frac{2\left(1-\frac{r}{s}\right)}{\beta A\frac{r}{s}}-1\right)}{\left(\frac{\frac{r}{s}+1}{2\frac{r}{s}}\right)}}\right\}$$

Equation (A.8)

For triangular wing-body combinations with no afterbody extending aft of the wing trailing edge, subsonic leading edges and $\beta \geq \frac{2\left(1-\frac{r}{s}\right)}{\frac{r}{s}A}$, $K_{B(W)}$ is:

$$K_{B(W)} = \frac{8\,E\left(\sqrt{1-\left(\frac{\beta A}{4}\right)^2}\right)}{\pi^2\left(\frac{s}{r}-1\right)}\left\{\sqrt{\frac{\beta A}{4}}\tan^{-1}\sqrt{\frac{4}{\beta A}}-\frac{\left(\frac{\beta A}{4}\right)}{\left[\left(\frac{\beta A}{4}\right)+1\right]}\right\} \tag{A.9}$$

For triangular wing-body combinations with no afterbody extension aft of the wing trailing edge, supersonic leading edges and $\beta \geq \frac{2\left(1-\frac{r}{s}\right)}{\frac{r}{s}A}$, $K_{B(W)}$ is: (See BKBWSPNA)

$$K_{B(W)} = \frac{2\left(\frac{\beta A}{4}\right)}{\pi\left(\frac{s}{r}-1\right)}\left[\frac{\pi}{2}-\frac{\left(\frac{\beta A}{4}\right)}{\sqrt{\left(\frac{\beta A}{4}\right)^2-1}}\cos^{-1}\left(\frac{4}{\beta A}\right)\right] \tag{A.10}$$

For triangular wing-body combinations with no afterbody extension aft of the wing trailing edge, supersonic leading edges and $\beta < \dfrac{2\left(1-\frac{r}{s}\right)}{\frac{r}{s}A}$, $K_{B(W)}$ is:

$$K_{B(W)} = \frac{\beta A}{\pi\sqrt{\left(\frac{\beta A}{4}\right)^2 - 1}}\left(\frac{\frac{r}{s}}{1-\frac{r}{s}}\right)^2\left[\left(\frac{\frac{r}{s}+1}{2\frac{r}{s}}\right)^2\cos^{-1}\left(\frac{\frac{r}{s}\frac{\beta A}{2}+\frac{4\left(1-\frac{r}{s}\right)}{\beta A}}{\frac{r}{s}+1}\right) - \frac{\left(1-\frac{r}{s}\right)^2}{4\left(\frac{r}{s}\right)^2}\cos^{-1}\left(\frac{4}{\beta A}\right) + \sqrt{\left(\frac{\beta A}{4}\right)^2 - 1}\left[\frac{\left(1-\frac{r}{s}\right)^2}{\beta A\left(\frac{r}{s}\right)^2}\sin^{-1}\left(\frac{\beta A\frac{r}{s}}{2\left(1-\frac{r}{s}\right)}\right) - \cosh^{-1}\left(\frac{2\left(1-\frac{r}{s}\right)}{\beta A\frac{r}{s}}\right)\right]\right]$$

Equation (A.11)

Note: BIGKBWA selects the correct equation to be used for triangular wing-body combinations with afterbodies. BIGKBWNA does the same for combinations with no afterbodies.

Figure A-1 plots values of the lift coefficient for slender-body theory.

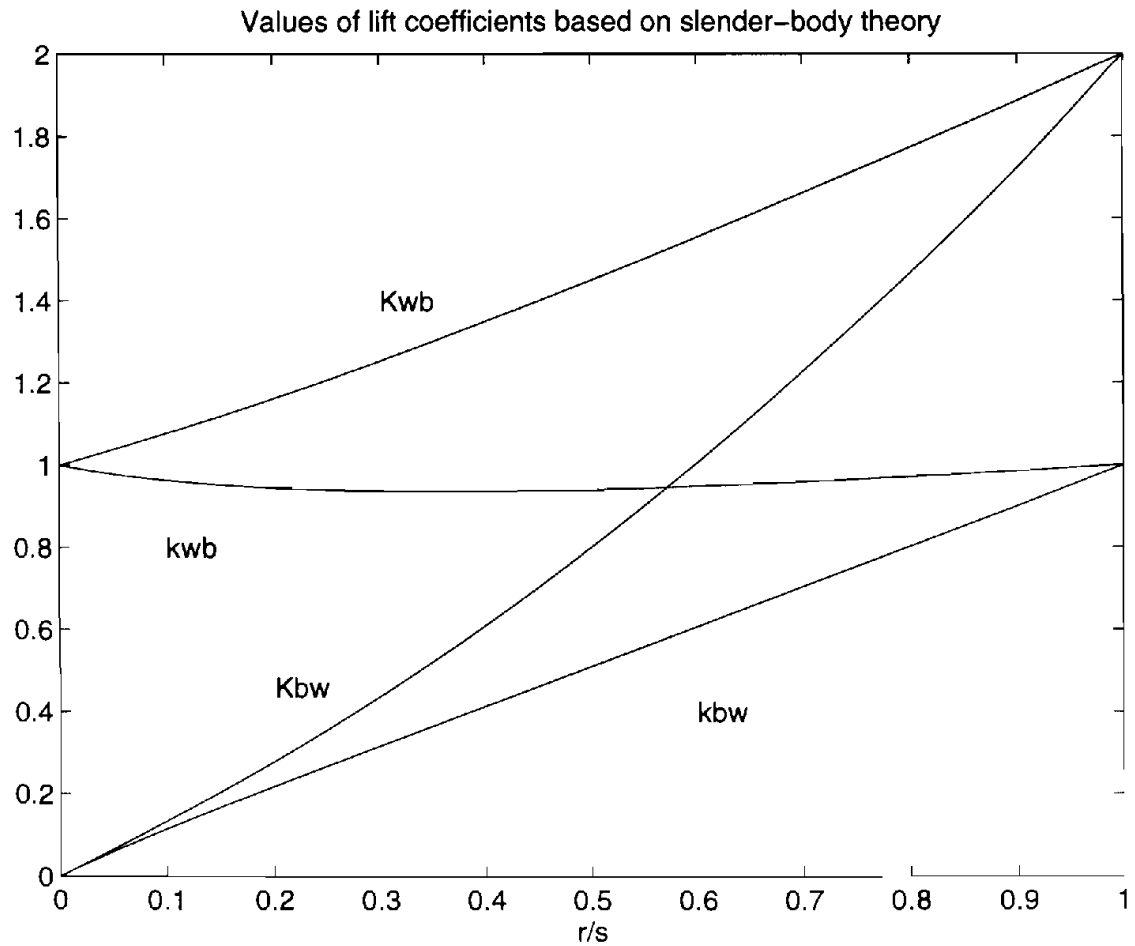


Figure A-1. Lift Coefficient Versus Radius-Semispan Ratio

Where:

$Kbw = K_{B(W)}$

$Kwb = K_{W(B)}$

$kbw = k_{B(W)}$

$kwb = k_{W(B)}$

APPENDIX B. MISSILE DESIGN TOOLBOX CODE

Appendix B consists of a print out of all the m-files of MSLDSN arranged alphabetically. Some minor editing was required to fit the format of the appendix.

AAMBSLN

```
function baseline = aambsln(diameter,length,range,dsplyoff)
%  AAMBSLN Calculates the baseline parameters for an air-to-air missile
%     using regression formulas.
%
%           baseline = AAMBSLN(diameter,length,range,dsplyoff)
%
%           diameter    = missile diameter (FT)
%           length      = missile length (FT)
%           range       = missile range(NM)
%           dsplyoff    = enter any numerical value to turn off the
%                         display summary sent to the screen
%
%           baseline    = [weight length wgtfinal lenfinal gcwt
%                          gclen whwt whlen pwt plen wfwt]
%
%           weight      = initial weight estimate
%           length      = initial length estimate
%           wgtfinal    = total weight based on sum of subsections
%           lenfinal    = total length based on sum of subsections
%           gcwt        = guidance/control subsection weight
%           gclen       = guidance/control subsection length
%           whwt        = warhead subsection weight
%           whlen       = warhead subsection length
%           pwt         = propulsion subsection weight
%           plen        = propulsion subsection length
%           wfwt        = total wing/fin weight

%     Turn the display summary on or off

if nargin < 4
     dsplyoff = 0;
else
     dsplyoff = 1;
end

%     Abbreviations
%
%     AR          Aspect Ratio
%     gclen       Guidance/Control Length
%     gcvol       Guidance/Control Volume
%     gcwt        Guidance/Control Weight
%     lenfinal    Total Missile Final Length
%     plen        Propulsion Length
%     pvol        Propulsion Volume
%     pwt         Propulsion Weight
%     SWP         Sweep Angle (DEG)
%     TR          Taper Ratio
%     wfwt        Wing/Fin Weight
%     wgtfinal    Total Missile Final Weight
%     whlen       Warhead Length
%     whvol       Warhead Volume
%     whwt        Warhead Weight

%     Estimate initial weight and density

volume = pi*diameter^2*length/4;

weight = 142.2*volume^0.74;
```

```
density = weight/volume;

%     Calculate subsection weights

pwt = -284.9+633.6*diameter-0.105*weight+0.949*density;

gcwt = 117.6*diameter+1.6*range-0.14*density;

whwt = 0.1*density-0.2*range+0.2*weight-2.4*length;

if range < 20
      SWP   = 45;
      AR    = 2.07;
      TR    = 0.63;
      wfwt  = 5.4+0.005*weight-0.2*SWP+11.1*TR;
      if wfwt < 0
            wfwt = 5.63;
      end
elseif range <= 50
      SWP   = 55;
      AR    = 2.38;
      wfwt  = -89.8+0.03*weight+0.99*SWP+13*AR;
      if wfwt < 0
            wfwt = 6.305;
      end
else
      SWP = 84;
      AR    = 0.56;
      wfwt  = 1.3*AR+0.1*SWP+0.0006*weight;
      if wfwt < 0
            wfwt = 10.00;
      end
end

wfwt = 8*wfwt;

wgtfinal = pwt+gcwt+whwt+wfwt;

%     Calculate subsection lengths based on diameter and subsection
weights

pvol = (pwt-2.7)/112;

plen = 4*pvol/pi/diameter^2;

gcvol = (gcwt/83.9)^1.5873;

gclen = 4*gcvol/pi/diameter^2;

whvol = (whwt/103.9)^1.2821;

whlen = 4*whvol/pi/diameter^2;

lenfinal = plen+gclen+whlen;

baseline = [weight length wgtfinal lenfinal gcwt gclen whwt];
baseline = [baseline whlen pwt plen wfwt];

%     Display summary data
```

```
if dsplyoff == 0

      disp(' ')
      disp('   Initial Estimate')
      disp('   Weight      Length')
      disp('     lb          ft')
      disp(baseline(1:2))
      disp('          Total          Guidance/Control          Warhead')
      disp('   Weight      Length   Weight     Length     Weight    Length')
      disp('     lb          ft       lb         ft         lb          ft')
      disp(baseline(3:8))
      disp('        Propulsion     Wing/Fin')
      disp('   Weight      Length   Weight')
      disp('     lb          ft       lb  ')
      disp(baseline(9:11))

END
```

AAMDUS

```
function data=aamdus()
%  AAMDUS Provides a data summary of U. S. air-to-air missiles
%
%                  data = aamdus
%
%    NAME       RANGE SPEED  LENGTH DIAMETER WEIGHT VOLUME  DENSITY
%                NM    MACH   FEET    FEET     LB    FT^3   LB/FT^3
%_____________________________________________________________________
%AMRAAM          35     4      12      0.6      339   3.39    99.92
%PHOENIX        100     2.5    11.8    0.7     1030  15.95    64.56
%SIDEWINDER      2     5       13      1.25     189   1.30   145.13
%SPARROW III     50     1.7     5      0.23     508   4.54   111.87
%STINGER         3     2       9.4    0.42      35   0.21   168.49
%
%  See also AAMDUSGC, AAMDUSP, AAMDUSWF, AAMDUSWH and AAMDWW.

%      Data from Naval Postgraduate School Master's Thesis
%      "Missile Total and Subsection Weight and Size Estimation
%      Equations"
%      by John B. Nowell Jr., June, 1992

data =[ 35     4      12       0.6       339    3.39     99.92
       100     2.5    11.8     0.7      1030   15.95     64.56
         2     5      13       1.25      189    1.30    145.13
        50     1.7     5       0.23      508    4.54    111.87
         3     2       9.4     0.42       35    0.21    168.49];
```

AAMDUSGC

```
function data=aamdusgc()
%  AAMDUSGC Provides the data for the guidance/control subsection of
%     U.S. air-to-air missiles.
%
%                  data = aamdusgc
%
%    NAME      LENGTH   DIAMETER   L/D   WEIGHT   VOLUME   DENSITY     WEIGHT
%               FEET      FEET             LB      FT^3    LB/FT^3    FRACTION
%_____________________________________________________________________________
%AMRAAM         5.88      0.60     9.8    120      1.66     72.33       0.355
%PHOENIX        6.96      1.25     5.57   300      8.54     35.12       0.291
%SIDEWINDER     2.04      0.42     4.86    30      0.28    104.4        0.156
%SPARROW III    4.13      0.70     5.9    134      1.59     84.56       0.264
%STINGER        1.19      0.23     5.17    14      0.05    283.2        0.400
%
%  See also AAMDUS,AAMDUSP, AAMDUSWF, AAMDUSWH and AANDWW.

%     Data from Naval Postgraduate School Master's Thesis
%     "Missile Total and Subsection Weight and Size Estimation
Equations"
%     by John B. Nowell Jr., June, 1992

data = [5.88      0.60     9.8   120      1.66     72.33      0.355
        6.96      1.25     5.57  300      8.54     35.12      0.291
        2.04      0.42     4.86   30      0.28    104.4       0.156
        4.13      0.70     5.9   134      1.59     84.56      0.264
        1.19      0.23     5.17   14      0.05    283.2       0.400];
```

AAMDUSP

```
function data=aamdusp()
%  AAMDUSP Provides the data for the propulsion subsection of U.S. air-
%     to-air missiles.
%
%                  data = aamdusp
%
%    NAME       LENGTH   DIAMETER   L/D   WEIGHT   VOLUME   DENSITY     WEIGHT
%                FEET      FEET              LB      FT^3    LB/FT^3    FRACTION
%_____________________________________________________________________________
%AMRAAM          4.89      0.60     8.15    154     1.38     111.67       0.455
%PHOENIX         3.32      1.25     2.66    460     4.07     112.91       0.447
%SIDEWINDER      5.83      0.42    13.88     99     0.81     122.57       0.524
%SPARROW III     4.95      0.70     7.07    211     1.90     110.92       0.416
%STINGER         3.25      0.23    14.13     17     0.14     125.90       0.486
%
%  See also AAMDUS, AAMDUSGC, AAMDUSWF, AAMDUSWH and AAMDWW.

%     Data from Naval Postgraduate School Master's Thesis
%     "Missile Total and Subsection Weight and Size Estimation
%     Equations"
%     by John B. Nowell Jr., June, 1992

data = [4.89     0.60    8.15   154    1.38    111.67     0.455
        3.32     1.25    2.66   460    4.07    112.91     0.447
        5.83     0.42   13.88    99    0.81    122.57     0.524
        4.95     0.70    7.07   211    1.90    110.92     0.416
        3.25     0.23   14.13    17    0.14    125.90     0.486];
```

AAMDUSWF

```
function data=aamduswf()
%  AAMDUSWF Provides data for wing/fin of U.S. air-to-air missiles.
%
%                  data = aamduswf
%
%    NAME          WEIGHT     TAPER     ASPECT     SWEEP ANGLE
%                    LB       RATIO     RATIO        DEGREES
%_____________________________________________________________
%AMRAAM             2.71       0.27      2.26          55
%PHOENIX           10.00       0.00      0.56          84
%SIDEWINDER         5.63       0.63      2.07          45
%SPARROW III        9.90       0.19      2.50          55
%
%Weight is for the weight of a single wing or fin.
%
%  See also AAMDUS, AAMDUSGC, AAMDUSP, AAMDUSWH and AAMDWW.

%      Data from Naval Postgraduate School Master's Thesis
%      "Missile Total and Subsection Weight and Size Estimation
%      Equations"
%      by John B. Nowell Jr., June, 1992

data = [ 2.71       0.27      2.26        55
        10.00       0.00      0.56        84
         5.63       0.63      2.07        45
         9.90       0.19      2.50        55];
```

AAMDUSWH

```
function data=aamduswh()
%  AAMDUSWH Provides the data for the warhead subsection of U.S.
%     air-to-air missiles.
%
%                  data = aamduswh
%
%    NAME       LENGTH   DIAMETER   L/D   WEIGHT   VOLUME   DENSITY     WEIGHT
%                FEET      FEET              LB      FT^3   LB/FT^3    FRACTION
%_____________________________________________________________________________
%AMRAAM          0.92      0.60    1.53     44      0.26     170.50      0.131
%PHOENIX         2.68      1.25    2.14    184      3.29      55.95      0.179
%SIDEWINDER      1.13      0.42    2.69     28      0.16     178.86      0.148
%SPARROW III     1.32      0.70    1.89     85      0.51     167.52      0.167
%STINGER         0.56      0.23    2.43      4      0.02     171.93      0.114
%
%  See also AAMDUS, AAMDUSGC, AAMDUSP, AAMDUSWF and AAMDWW.

%     Data from Naval Postgraduate School Master's Thesis
%     "Missile Total and Subsection Weight and Size Estimation
%     Equations"
%     by John B. Nowell Jr., June, 1992

data = [0.92     0.60    1.53     44      0.26     170.50      0.131
        2.68     1.25    2.14    184      3.29      55.95      0.179
        1.13     0.42    2.69     28      0.16     178.86      0.148
        1.32     0.70    1.89     85      0.51     167.52      0.167
        0.56     0.23    2.43      4      0.02     171.93      0.114];
```

AAMDWW

```
function data = aamdww()
%  AAMDWW  Provides a data summary of world wide air-to-air missiles.
%
%            data = aamdww
%
%     NAME        RANGE      COUNTRY      WEIGHT       VOLUME        DENSITY
%                 (NM)                      LB          FT^3         LB/FT^3
%_____________________________________________________________________________
%AMRAAM            35          USA          339          3.39          99.92
%ANAB              11          USSR         605          8.52          70.97
%APEX              19          USSR         704          3.39         207.13
%APHID              4          USSR         121          0.89         135.62
%ASH                9          USSR         860         13.35          64.41
%ASPIDE            40          ITALY        485          4.66         104.16
%ATOLL              3.5        USSR         154          1.16         133.21
%KUKRI              2          S. AFRICA    161          1.21         133.46
%MAA-1              5          BRASIL       198          1.81         109.61
%MAGIC              5          FRANCE       200          1.77         113.18
%PHOENIX          100          USA         1030         15.95          64.56
%PL-5B              8.6        CHINA        187          1.19         156.64
%PYTHON             8          ISREAL       264          1.92         137.20
%R.530              9.7        FRANCE       423          6.87          61.57
%SHAFRIR            2.7        ISREAL       205          1.61         127.33
%SIDEWINDER         2          USA          189          1.30         145.13
%SKYFLASH          27          UK           425          4.62          92.03
%SPARROW III       50          USA          508          4.54         111.87
%STINGER            3          USA           35          0.21         168.49
%SUPER 530         22          FRANCE       550          7.38          74.53
%
%  See also AAMDUS, AAMDUSGC, AAMDUSP, AAMDUSWF and AAMDUSWH.

%      Data from Naval Postgraduate School Master's Thesis
%      "Missile Total and Subsection Weight and Size Estimation
%      Equations"
%      by John B. Nowell Jr., June, 1992

data = [ 35    339   3.39    99.92
         11    605   8.52    70.97
         19    704   3.39   207.13
          4    121   0.89   135.62
          9    860  13.35    64.41
         40    485   4.66   104.16
          3.5  154   1.16   133.21
          2    161   1.21   133.46
          5    198   1.81   109.61
          5    200   1.77   113.18
        100   1030  15.95    64.56
          8.6  187   1.19   156.64
          8    264   1.92   137.20
          9.7  423   6.87    61.57
          2.7  205   1.61   127.33
          2    189   1.30   145.13
         27    425   4.62    92.03
         50    508   4.54   111.87
          3     35   0.21   168.49
         22    550   7.38    74.53];
```

AREAPOGV

```
function Ap = areapogv(d,ln)
%  AREAPOGV Returns the planform area of an ogival nose.
%
%            Ap = areapogv(d,ln)
%
%            Ap     = planform area
%            d      = diameter
%            ln     = length of the nose

R = d/4+ln^2/d;
a = R-d/2;
Ap = ln*sqrt(R^2-ln^2)+R^2*asin(ln/R)-2*a*ln;
```

ATMSCOFV

```
function cofv = atmscofv(altitude,iunits)
%  ATMSCOFV Calculates the coefficient of viscosity (lbm/ft-sec) of a
%     standard atmosphere at a given altitude.
%
%           cofv = atmscofv(altitude)
%
%     English is the default unit type.  If Metric units (kg/sec-m) are
%     required, use:
%           cofv = atmscofv(altitude,1)
%
%     See also ATMSDENS, ATMSPRSS, ATMSSOFS, ATMSTEMP and STAN62.

[m,n] = size(altitude);
cofv = zeros(m,n);

l = length(altitude);

for k = 1:l
      if nargin < 2
            data = stan62(altitude(k));
            cofv(k)  = data(6);
      else
            data = stan62(altitude(k),1);
            cofv(k)  = data(6);
      end
end
```

ATMSDENS

```
function rho = atmsdens(altitude,iunits)
%  ATMSDENS Calculates the density (slug/ft^3) of a standard atmosphere
%      at a given altitude.
%
%            rho = atmsdens(altitude)
%
%      English is the default unit type.  If Metric units (kg/m^3) are
%      required, use:
%            rho = atmsdens(altitude,1)
%
%      See also ATMSCOFV, ATMSPRSS, ATMSSOFS, ATMSTEMP and STAN62.

[m,n] = size(altitude);
rho = zeros(m,n);

l = length(altitude);

for k = 1:l
      if nargin < 2
            data = stan62(altitude(k));
            rho(k)  = data(3);
      else
            data = stan62(altitude(k),1);
            rho(k)  = data(3);
      end
end
```

ATMSPRSS

```
function prss = atmsprss(altitude,iunits)
%  ATMSPRSS Calculates the temperature (lb/ft^2) of a standard
%     atmosphere at a given altitude.
%
%           prss = atmsprss(altitude)
%
%     English is the default unit type.  If Metric units (kg/m^2) are
%     required, use:
%           prss = atmsprss(altitude,1)
%
%     See also ATMSCOFV, ATMSDENS, ATMSSOFS, ATMSTEMP and STAN62.

[m,n] = size(altitude);
prss = zeros(m,n);

l = length(altitude);

for k = 1:l
      if nargin < 2
            data = stan62(altitude(k));
            prss(k)  = data(4);
      else
            data = stan62(altitude(k),1);
            prss(k)  = data(4);
      end
end
```

ATMSSOFS

```
function sofs = atmssofs(altitude,iunits)
%  ATMSSOFS Calculates the speed of sound (ft/s) of a standard
%     atmosphere at a given altitude.
%
%           sofs = atmssofs(altitude)
%
%     English is the default unit type.  If Metric units (m/s) are
%     required, use:
%           sofs = atmssofs(altitude,1)
%
%     See also ATMSCOFV, ATMSDENS, ATMSPRSS, ATMSTEMP and STAN62.

[m,n] = size(altitude);
sofs = zeros(m,n);

l = length(altitude);

for k = 1:l
      if nargin < 2
            data = stan62(altitude(k));
            sofs(k)  = data(2);
      else
            data = stan62(altitude(k),1);
            sofs(k)  = data(2);
      end
end
```

ATMSTEMP

```
function temp = atmstemp(altitude,iunits)
%  ATMSTEMP Calculates the temperature (R) of a standard atmosphere at a
%     given altitude.
%
%           temp = atmstemp(altitude)
%
%     English is the default unit type.  If Metric units (K) are
%     required, use:
%           temp = atmstemp(altitude,1)
%
%     See also ATMSCOFV, ATMSDENS, ATMSPRSS, ATMSSOFS and STAN62.

[m,n] = size(altitude);
temp = zeros(m,n);

l = length(altitude);

for k = 1:l
      if nargin < 2
            data = stan62(altitude(k));
            temp(k)  = data(5);
      else
            data = stan62(altitude(k),1);
            temp(k)  = data(5);
      end
end
```

BDYCLA

```
function CLaB = bdycla(d,l,ln,M,alpha)
%  BDYCLA Calculates the body CLa for a body with an ogival nose
%     including the nonlinear angle of attack region.
%
%           CLaB = bdyclasb(d,l,ln,M,alpha)
%
%           CLaB  = lift-curve-slope for the body based on base area
%           l     = body length
%           d     = body base diameter
%           ln    = nose length
%           M     = Mach number
%           alpha = angle of attack (RAD).  If no value is entered a
%                   of alpha = 0 is assumed.

if nargin < 5
      alpha = 0;
end

alpha = alpha*pi/180;

Sref = pi*(d/2)^2;

cdc = sscfdccc(alpha,M);

Ap = d*(l-ln)+areapogv(d,ln);

if M <= 1

      n = dragfctr(l/d);

      CLaB = k2mk1(l/d)*2+Ap*n*cdc*alpha/Sref;

else

      CLaB = k2mk1(l/d)*2+cdc*Ap*alpha/Sref;

end
```

BIGKBW

```
function bigkbw()
%  BIGKBW:
%
%      Seven different m-files are available for calculating Kb(w) for
%      triangular wings.
%
%      for subsonic condition use BIGKBWSB
%      else
%
%                                      afterbody              no afterbody
%
%      supersonic leading edge         BKBWSPA                 BKBWSPNA
%
%      subsonic leading edge           BKBWSBA                 BKBWSBNA
%
%
%      BIGKBWA selects the m-file for triangular wings with afterbodies.
%      BIGKBWNA selects for triangular wings with no afterbody.
```

BIGKBWA

```
function Kbw = bigkbwa(M,A,r,s)
%  BIGKBWA Returns the value for Kb(w) interference factor for a
%     triangular wing with an afterbody.
%
%                  Kbw = bigkbwa(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = body radius
%                  s = maximum semispan of wing + r

%     From NACA 1307

b = sqrt(abs(M^2-1));

if b*A <= 0
      Kbw = bigkbwsb(r,s);
else
      if b*A/4 < 1
            Kbw = bkbwsba(M,A,r,s);
      else
            Kbw = bkbwspa(M,A,r,s);
      end
end
```

BIGKBWA

```
function Kbw = bigkbwna(M,A,r,s)
%  BIGKBWA Returns the value for Kb(w) interference factor for a
%     triangular wing with no afterbody.
%
%                  Kbw = bigkbwna(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = body radius
%                  s = maximum semispan of wing + r

%     From NACA 1307

b = sqrt(abs(M^2-1));

if b*A <= 0
      Kbw = bigkbwsb(r,s);
else
      if b*A/4 < 1
            Kbw = bkbwsbna(M,A,r,s);
      else
            Kbw = bkbwspna(M,A,r,s);
      end
end
```

BIGKBWSB

```
function Kbw = bigkbwsb(r,s)
%  BIGKBWSB Calculates KB(W) the lift of the body in the presence of the
%     wing using slender body theory.
%
%                  Kbw = bigkbwsb(r,s)
%
%                  r = body radius
%                  s = r + semispan
%
%     BGKBWSB is valid if b*A <= 0
%
%                  b     = sqrt(abs(M^2-1))
%                  M     = Mach Number
%                  A     = aspect ratio

%     David A. Ekker from NACA 1307

t = r/s;

if t >= 1
      Kbw = 2;
else
      t1 = 1/t;
      Kbw = (1+t^4)*(0.5*atan(0.5*(t1-t))+pi/4)-t^2*((t1-t)+2*atan(t));
      Kbw = ((1-t^2)^2-2*Kbw/pi)/(1-t)^2;
end
```

BIGKWB

```
function Kwb = bigkwb(r,s)
%  BIGKWB Calculates KW(B) the lift of the wing in the presence of the
%     body for variable angle of attack using slender body theory.
%
%                  Kwb = bigkwb(r,s)

%     Written by David A. Ekker derived from NACA 1307

t = r/s;

if t >= 1
      Kwb = 2;
else
      t1 = 1/t;
      Kwb = (1+t^4)*(0.5*atan(0.5*(t1-t))+pi/4)-t^2*((t1-t)+2*atan(t));
      Kwb = 2*Kwb/pi/(1-t)^2;
end
```

BKBWSBA

```
function Kbw = bkbwsba(M,A,r,s)
%  BKBWSBA Calculates the lift on the body due to the wing Kb(w) using
%     theory from NACA 1307 for triangular wing body combinations with
%     subsonic leading edges and an afterbody behind the wing.
%
%                  Kbw = bkbwsba(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = body radius
%                  s = maximum semispan of wing + r
%
%
%     BKBWSBA is valid if b*A > 0
%
%                  b = sqrt(abs(M^2-1))

%     David A. Ekker

b = sqrt(abs(M^2-1));
t = r/s;
z = b*A/4;
[K,E] = ellipke(sqrt(1-z^2));

t1 = 2*(1+z)*t/(1-t)+1;
t2 = z^2/(z+1)^2/2;

Kbw = 2*z^2*(t/(1-t))^2*atanh(sqrt(1/t1))+(b*A/(b*A+4))^2;
Kbw = t2*(t1^1.5+t1^0.5)-Kbw;
Kbw = 8*E*Kbw/z^2/pi^2;

Kbw = round(Kbw*100)/100;
```

BKBWSBNA

```
function Kbw = bkbwsbna(M,A,r,s)
%  BKBWSBNA Calculates the lift on the body due to the wing Kb(w) using
%     theory from NACA 1307 for triangular wing body combinations with
%     subsonic leading edges and no afterbody behind the wing.
%
%                  Kbw = bkbwsbna(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = body radius
%                  s = maximum semispan of wing + r
%
%
%     BKBWSBNA is valid if b*A > 0
%
%                  b = sqrt(abs(M^2-1))

%     David A. Ekker from NACA 1307

b = sqrt(abs(M^2-1));
t = r/s;
z = b*A/4;
[K,E] = ellipke(sqrt(1-z^2));
t2 = z+1;

F = 2*(1-t)/t/A;

if b < F
      t3 = (t+1)/2/t;
      t4 = (1-t)^2/t^2;
      t1 = 2*(1-t)/b/A/t;
      Kbw = atanh(sqrt(z*(t1-1)/t3))*t2/sqrt(z)+t1^2*z^1.5;
      Kbw = t4*t2*(atan(sqrt(1/z))-atan(sqrt((t1-1)/t3)))/b/A-Kbw;
      Kbw = Kbw+t3*sqrt((t1-1)*t3);
      Kbw = 16*z^0.5*E*Kbw/pi^2/t2/t4;
else
      z1 = sqrt(z);
      Kbw = (8*E/pi^2/(1/t-1))*(z1*atan(1/z1)-z/t2);
end
```

BKBWSPA

```
function Kbw = bkbwspa(M,A,r,s)
%  BKBWSPA Calculates the lift on the body due to the wing Kb(w) using
%     theory from NACA 1307 for triangular wing body combinations with
%     supersonic leading edges and an afterbody behind the wing.
%
%                  Kbw = bkbwspa(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = body radius
%                  s = maximum semispan of wing + r
%
%
%     BKBWSPA is valid if b*A > 0
%
%                  b = sqrt(abs(M^2-1))

%     David A. Ekker

b = sqrt(abs(M^2-1));
t = r/s;
z = b*A/4;

t1 = 2*(1+z)*t/(1-t)+1;
t2 = sqrt(z^2-1);
t3 = 1+z;
t4 = b*A*t3*t/2/(1-t);

Kbw = t2*b*A*(t/(1-t))^2*acosh(1+2*(1-t)/b/A/t)+t2/t3+z*acos(1/z)/t3;
Kbw = t2*sqrt(1+b*A*t/(1-t))/t3-Kbw;
Kbw = (b*A/(b*A+4))*t1^2*acos((1+t4)/(z+t4))+Kbw;
Kbw = Kbw/pi/t2;

Kbw = round(Kbw*100)/100;
```

BKBWSPNA

```
function Kbw = bkbwspna(M,A,r,s)
%  BKBWSPNA Calculates the lift on the body due to the wing Kb(w) using
%     theory from NACA 1307 for triangular wing body combinations with
%     supersonic leading edges and no afterbody behind the wing.
%
%                  Kbw = bkbwspna(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = body radius
%                  s = maximum semispan of wing + r
%
%
%     BKBWSPNA is valid if b*A >0
%
%                  b = sqrt(abs(M^2-1))

%     David A. Ekker from NACA 1307

b = sqrt(abs(M^2-1));
t = r/s;
z = b*A/4;

t1 = (1-t)/t;
t2 = sqrt(z^2-1);

F = 2*t1/A;

if b < F
      Kbw = t2*acosh(2*t1/b/A);
      Kbw = Kbw+t1^2*acos(1/z)/4;
      Kbw = t1^2*t2*asin(b*A/t1/2)/b/A-Kbw;
      Kbw = ((t+1)/2/t)^2*acos((t*A*b/2+4*(1-t)/A/b)/(t+1))+Kbw;
      Kbw = b*A*Kbw/pi/t2/t1^2;
else
      Kbw = (2*z/pi/(1/t-1))*(pi/2-z*acos(1/z)/t2);
end
```

BSDRGCF

```
function bdcf = bsdrgcf(Mach)
%  BSDRGCF Calculates the additional base drag to be added to the base
drag
%      at Mach = 0.6 for a subsonic body with the base diameter equal to
%      the body diameter.
%
%            bdcf = bsdrgcf(Mach)
%
%            Mach = Mach number

p = [-3.330968357195938e+004;2.103304015137906e+005;
     -5.784664805761514e+005;9.050874924571224e+005;
     -8.811731632140930e+005;5.466276717656434e+005;
     -2.109994072780218e+005;4.633522170518622e+004;
     -4.431983251061754e+003];

bdcf = polyval(p,Mach);
```

BSDRGSP

```
function cpb = bsdrgsp(Mach)
%  BSDRGSP Returns the base pressure coefficient for a body in the
%     supersonic region with a base to body diameter ratio = 1.  Base
%     drag coefficient is CDb = -Cpd.
%
%           cpb = bsdrgsp(Mach)
%
%           Mach  = Mach number


p = [-2.903303569625142e-005; 8.917615745536690e-004;-
1.171998069776006e-002;
      8.580630708865064e-002;-3.811230276820538e-001;
1.044387398043697e+000;
     -1.697460661376502e+000; 1.405711843047642e+000;-
2.362628974278631e-001];

cpb = -polyval(p,Mach);
```

CDBBODY

```
function CDB = cdbbody(d,l,ln,Mach,atr)
%  CDBBODY Calculates the body drag due to angle of attack for a
%     body with ogival nose.
%
%           CDB = cdbbody(d,l,ln,Mach,atr)
%
%           d     = missile diameter
%           l     = length of body
%           ln    = length of the nose
%           Mach  = Mach number
%           atr   = trim angle of attack (RAD)

Sref = pi*(d/2)^2;

cdc = sscfdccc(atr,Mach);

Ap = d*(l-ln)+areapogv(d,ln);

if Mach <= 1
      n = dragfctr(l/d);
      CDB = 2*k2mk1(l/d)*atr^2+n*cdc*Ap*atr^3/Sref;
else
      CDB = 2*k2mk1(l/d)*atr^2+cdc*Ap*atr^3/Sref;
end
```

CDLCOMP

```
function corr = cdlcomp(bw,Sw,Mach)
%  CDLCOMP Returns the correlation function of the drag due to lift of a
%     composite planform.  The input is simplified for a triangular
%     planform.
%
%           corr = cdlcomp(bw,Sw,Mach)
%
%           corr  = correlation function
%           bw    = wing span
%           Sw    = wing area
%           Mach  = Mach number

beta = sqrt(Mach^2-1);

Aw = bw^2/Sw;

p = [-1.131676150597421e+000; 7.455524067331654e+000;-
1.959790137335054e+001;
      2.617725155474472e+001;-1.909733706084593e+001;
8.093378527305321e+000;
     -1.217830853307086e+000; 5.365069933169143e-001];

corr = polyval(p,beta*Aw/4);
```

CDLWING

```
function cdl = cdlwing(bw,Sw,Mach,atr)
%  CDLWING Calculates the coefficient of drag due to lift of a wing
%     based on the surface area of the wing.
%
%           cdl = cdlwing(bw,Sw,Mach,atr)
%
%           bw    = wing span (FT)
%           Sw    = wing area (FT^2)
%           Mach  = Mach number
%           atr   = trim angle of attack (RAD)

Aw = bw^2/Sw;

if Mach == 0
      Mach = 0.0001;
end

if atr == 0
      cdl = 0;
end

if Mach <= 1
      kappaw = 0.85;
      lc2w = atan(2/Aw);
      CLaW = clawsub(Mach,Aw,lc2w,kappaw);
      cdl = CLaW*atr^2/1.1;
else
      betaw = sqrt(Mach^2-1);
      m = Aw/4;

      if m*betaw >= 1
            cdl = 4*atr^2/betaw;
      else
            corr = cdlcomp(bw,Sw,Mach);
            CLaW = clawsup(Mach,Aw);
            cdl = 3*(CLaW*atr)^2*corr/pi/Aw;
      end
end
```

CDOBODY

```
function cdo = cdobody(d,l,ln,Mach,alt,nf,pwroff)
%  CDOBODY Calculates the cdo for a cylindrical body with a tangent
%     ogive nose.  No base pressure drag is calculated.  Turbulent
%     flow is assumed.
%
%           cdo = cdobody(d,l,ln,Mach,alt,nf,pwroff)
%
%           d      = body diameter (FT)
%           l      = body length (FT)
%           ln     = nose length (FT)
%           Mach   = Mach number
%           alt    = Altitude (FT)
%           nf     = friction factor.  the friction coefficient
%                    is multiplied by nf.  If no value or 0 is entered
%                    nf = 1.1
%           pwroff      = any value ~=0 entered for pwroff will cause
base
%                    drag added to the body drag

if nargin < 6
      nf = 1.1;
end

if nf == 0
      nf = 1.1;
end

if Mach == 1
      Mach = 0.999999;
end

Sref = pi*d^2/4;

%  Calculate the wetted surface area of the body less the base
Swet = surfogv1(d,ln)+(1-ln)*pi*d;

if Mach <= 0
      cdo = 0;
elseif Mach <=1
      if Mach > 0.6
            Mach1 = 0.6;
      else
            Mach1 = Mach;
      end
      cf = nf*cfturbfp(l,Mach1,alt);
      CDF = cf*Swet/Sref;
      CDP = cf*(60/(l/d)^3+0.0025*l/d)*Swet/Sref;
      cdo = CDF+CDP;
else
      cf = nf*cfturbfp(l,Mach,alt);
      CDW = wvdrgogv(d,ln,Mach);
      CDF = cf*Swet/Sref;
      CDP = 0;
      if Mach < 1.2
            cf = nf*cfturbfp(l,0.6,alt);
            CDPp6 = cf*(60/(l/d)^3+0.0025*l/d)*Swet/Sref;
            CDP = CDPp6*(1.2-Mach)/0.2;
      end
```

```
      cdo = CDF+CDW+CDP;
end

if nargin == 7
      if pwroff ~= 0
            if Mach <= 0
                  cdb = 0;
            elseif Mach <= 1
                  cdb = 0.029/sqrt(CDF);

                  if Mach > 0.6
                        cdb = cdb+bsdrgcf(Mach);
                  end
            else
                  cdb = -bsdrgsp(Mach);
            end

            cdo = cdo+cdb;
      end
end
```

CDOWBT

```
function cdomsl =
cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,Mach,alt,nf,pwroff)
%  CDOWBT Calculates the zero lift drag coeeficient for a wing-body-tail
%     combination with zero control deflection.
%
%cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,Mach,alt,nf,pwroff)
%
%           d     = body diameter (FT)
%           l     = body length (FT)
%           ln    = nose length (FT)
%           bw    = wing span (FT)
%           Sw    = exposed wing area (FT^2)
%           thickw= wing thickness ratio
%           bt    = tail span (FT)
%           St    = exposed tail area (FT^2)
%           thickt= tail thickness ratio
%           Mach  = Mach number
%           alt   = Altitude (FT)
%           nf    = friction factor.  the friction coefficient
%                   is multiplied by nf.  If no value or 0 is entered
%                   nf = 1.1
%           pwroff= any value ~=0 entered for pwroff will cause base
%                   drag added to the body drag

Sref = pi*d^2/4;

if nargin < 12
      nf = 1.1;
end

if nf == 0
      nf = 1.1;
end

if nargin == 13
      if pwroff ~= 0
            pwroff = 1;
      end
else
      pwroff = 0;
end

if Mach <= 0.95|Mach >= 1.1
      cdob = cdobody(d,l,ln,Mach,alt,nf,pwroff);
      cdow = cdowing(bw,Sw,thickw,Mach,alt,nf);
      cdot = cdowing(bt,St,thickt,Mach,alt,nf);
      cdomsl = cdob+2*cdow*Sw/Sref+2*cdot*St/Sref;
else
      %  Calculation of drag in the transonic region
      cd0p95 =
cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,.95,alt,nf,pwroff);
      cd01p1 =
cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,1.1,alt,nf,pwroff);
      dCdM = (cd01p1-cd0p95)/0.15;
      cdomsl = dCdM*(Mach-0.95)+cd0p95;
      if cdomsl > cd01p1
            cdomsl = cd01p1;
      end
end
```

CDOWING

```
function cdo = cdowing(bw,Sw,thick,Mach,alt,nf)
%  CDOWING Calculates the cdo for a triangular planform, double wedge
%     wing referenced to the area of the wing.
%
%           cdo = cdowing(bw,Sw,thick,Mach,alt,nf)
%
%           bw    = wing span (FT)
%           Sw    = exposed wing area (FT^2)
%           thick = wing thickness ratio
%           Mach  = Mach number
%           alt   = altitude (FT)
%           nf    = friction factor.  the friction coefficient
%                   is multiplied by nf.  If no value is entered
%                   nf = 1.1

if nargin < 6
      nf = 1.1;
end

if Mach == 1
      Mach = 0.999999;
end

MAC = (2/3)*2*Sw/bw;
SWP = atan(4*Sw/bw^2);

if Mach == 0
      cdo = 0;
elseif Mach <= 1
      cf = nf*cfturbfp(MAC,Mach,alt);
      cdo = 2*cf*(1+1.2*thick+60*thick^4);
else
      cf = nf*cfturbfp(MAC,Mach,alt);
      beta = sqrt(Mach^2-1);
      B = 4;                                  %  Double wedge wing
      m = 1/tan(SWP*pi/180);

      %  Calculate wave drag
      if m*beta >= 1                          %  Supersonic leading edge
            cdw = B*thick^2/beta;
      else                                %  Subsonic leading edge
            cdw = B*m*thick^2;
      end

      cdo = 2*cf+cdw;
end
```

CDTRIM

```
function cdtr =
cdtrim(d,l,ln,bw,Sw,thckw,bt,St,thckt,Mach,alt,atr,dtr,nf,pwroff)
%  CDTRIM calculates the coefficient of drag for a specified trim angle
%     of attack and trim deflection angle.
%
% cdtr=cdtrim(d,l,ln,bw,Sw,thckw,bt,St,thckt,Mach,alt,atr,dtr,nf,pwroff)
%
%            d      = body diameter (FT)
%            l      = body length (FT)
%            ln     = nose length (FT)
%            bw     = wing span (FT)
%            Sw     = exposed wing area (FT^2)
%            thckw  = wing thickness ratio
%            bt     = tail span (FT)
%            St     = exposed tail area (FT^2)
%            thckt  = tail thickness ratio
%            Mach   = Mach number
%            alt    = Altitude (FT)
%            atr    = trim angle of attack (RAD)
%            dtr    = trim tail deflection angle (RAD)
%            nf     = friction factor.  the friction coefficient
%                     is multiplied by nf.  If no value or 0 is entered
%                     nf = 1.1
%            pwroff= any value ~=0 entered for pwroff will cause base
%                     drag added to the body drag
atr = abs(atr);
dtr = abs(dtr);

if nargin < 14
      nf = 1.1;
      pwroff = 0;
elseif nargin < 15
      pwroff = 0;
end

if nf == 0
      nf = 1.1;
end

Sref = pi*d^2/4;

%  Calculate the CDo of the body, wing and vertical surfaces.
CDoB = cdobody(d,l,ln,Mach,alt,nf,pwroff);
CDoW = cdowing(bw,Sw,thickw,Mach,alt,nf);
CDoT = cdowing(bt,St,thickt,Mach,alt,nf);
CDoWB = CDoB+2*CDoW*Sw/Sref+CDoT*St/Sref;

%  Calculate the drag of the wing-body due to angle of attack
CDLW = cdlwing(bw,Sw,Mach,atr);
CDBa = cdbbody(d,l,ln,Mach,atr);
CDiWB = CDLW*Sw/Sref+CDBa;

%  Calculate horizontal tail contribution
Mt = 0.95*Mach;
CDLT = cdlwing(bt,St,Mt,dtr);
CDT = CDoT+CDLT;
CLd = cldwbt(d,bt,St,Mt)*Sref/St;
CLT = CLd*dtr;
```

```
nt = (Mt/Mach)^2;
CDTR = (CDT*cos(dtr)+CLT*sin(dtr))*St*nt/Sref;

cdtr = CDoWB+CDiWB+CDTR;
```

CDTRIMEX

```
%********************CDTRIMEX********************
%
%  CDTRIMEX calculates the coefficient drag due to lift of a missile
%     and compares it to the value of zero-lift coefficient alone
%     for a Mach range of 1.2 to 2.5.  The maximum difference
%     between the two values is returned.

Mach = 1.2:.1:2.5;

%  Enter missile data
d = 0.6;
l =12;
lcg = 6.5591;
ln = 2*d;
bw = 2.5844;
Sw = 2.3847;
lw = 5.5929;
bt=1.2751;
St=0.5805;
thickw = 0.04;
thickt = 0.04;
alt = 10000;
n = 1;
W = 339;

Sref = pi*d^2/4;

cdomsl = zeros(size(Mach));
cdtrlg = zeros(size(Mach));

for k = 1:length(Mach)
      cdomsl(k) = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,Mach(k),alt);

      Cma = cmawbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach(k));
      Cmd = cmdwbt(d,l,lcg,bt,St,Mach(k));
      CLa = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach(k));
      CLd = cldwbt(d,bt,St,Mach(k));

      qbar = dynmprss(Mach(k),alt);

      A = [Cma Cmd;CLa CLd];
      Y = [0;n*W/qbar/Sref];

      X=A\Y;
      atr = X(1);
      dtr = X(2);

      cdtrlg(k)
= cdtrim(d,l,ln,bw,Sw,thickw,bt,St,thickt,Mach(k),alt,atr,dtr);
end

[maxdiff,index] = max(abs(cdtrlg-cdomsl));
maxerlg = maxdiff/cdomsl(index)

n = 10;
cdtr10g = zeros(size(Mach));

for k = 1:length(Mach)
```

```
        Cma = cmawbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach(k));
        Cmd = cmdwbt(d,l,lcg,bt,St,Mach(k));
        CLa = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach(k));
        CLd = cldwbt(d,bt,St,Mach(k));

        qbar = dynmprss(Mach(k),alt);

        A = [Cma Cmd;CLa CLd];
        Y = [0;n*W/qbar/Sref];

        X=A\Y;
        atr = X(1);
        dtr = X(2);

        cdtr10g(k) =
cdtrim(d,l,ln,bw,Sw,thickw,bt,St,thickt,Mach(k),alt,atr,dtr);
end

[maxdiff,index] = max(abs(cdtr10g-cdoms1));
maxer10g = maxdiff/cdoms1(index)

clf
plot(Mach,cdtr1g,Mach,cdoms1,Mach,cdtr10g)
title('Zero-Lift Drag and Drag Due to Lift')
xlabel('Mach Number')
ylabel('Drag Coefficient')
text(1.6,0.7,'Load Factor = 1')
text(1.8,1,'Load Factor = 10')
text(1.3,0.5,'Zero Lift')
```

CFTURBFP

```
function cf = cfturbfp(chardim,Mach,altitude)
%  CFTURBFP Calculates the coefficient of friction for a flat plate in
%     turbulent flow, corrected for compressibility.
%
%          cf = cfturbfp(chardim,Mach,altitude)
%
%           chardim  = Characteristic dimension
%           Mach     = Mach number
%           altitude = altitude

Rn = reynolds(chardim,Mach,altitude);
cfi = 0.455/(log10(Rn))^2.58;

if Mach <= 1
      cf = cfi/(1+0.08*Mach^2);
else
      cf = cfi/(1+0.144*Mach^2)^0.65;
end
```

CLAMSLSB

```
function CLa = clamslsb(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha)
%  CLAMSLSB Returns the subsonic lift-curve-slope of a body-wing-tail
%     combination.  The assumptions include triangular planform
%     aerodynamic surfaces, an ogival nose with a cylindrical afterbody.
%     The lift-curve-slope is referenced to the area of the body
%     diameter.
%
%           cla = clamslsb(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha)
%
%           d     = missile diameter (FT)
%           l     = missile length (FT)
%           ln    = length of nose (FT)
%           bw    = wing span (FT)
%           Sw    = wing surface area (FT^2)
%           lw    = location of wing leading edge-body intersection
%                   measured from the tip of the nose (FT)
%           bt    = tail span (FT)
%           St    = tail surface area (FT^2)
%           Mach  = Mach number
%           alpha = assumed angle of attack (DEG).  This is required for
%                   calculations in the non-linear angle of attack
%                   region.  If no value is entered for alpha,
%                   alpha = 0 is assumed.

if nargin < 10
      alpha = 0;
end
alpha = alpha*pi/180;
Sref = pi*(d/2)^2;
%  Calculate nose-body lift-curve-slope
CLaN = bdycla(d,l,ln,Mach,alpha);
%  Calculate the wing parameters
Aw = bw^2/Sw;
sw = 0.5*(d+bw);
lc2w = atan(2/Aw);
lc4w = atan(3/Aw);
kappaw = 0.85;
CLaW = clawsub(Mach,Aw,lc2w,kappaw);
CLaW0 = clawsub(0,Aw,lc2w,kappaw);
Kbw = bigkbwsb(d/2,sw);
Kwb = bigkwb(d/2,sw);
%  Calculate the tail parameters
Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
At = bt^2/St;
st = 0.5*(d+bt);
lc2t = atan(2/At);
kappat = 0.85;
CLaT = clawsub(Mt,At,lc2t,kappat);
lH = 1-St/bt-(lw+Sw/bw);
deda = dedasub(Aw,bw,lc4w,0,lH,0);
dedaM = deda*CLaW/CLaW0;
Kbt = bigkbwsb(d/2,st);
Ktb = bigkwb(d/2,st);
CLa = CLaN+(Kbw+Kwb)*CLaW*Sw/Sref+(Kbt+Ktb)*CLaT*(1-dedaM)*nt*St/Sref;
```

CLAMSLSP

```
function CLa = clamslsp(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha)
%  CLAMSLSP Returns the supersonic lift-curve-slope of a body-wing-tail
%     combination.  The assumptions include triangular planform
%     aerodynamic surfaces, an ogival nose with a cylindrical afterbody.
%     The lift-curve-slope is referenced to the area of the body
%     diameter.
%
%           cla = clamslsp(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha)
%
%           d     = missile diameter (FT)
%           l     = missile length (FT)
%           ln    = length of nose (FT)
%           bw    = wing span (FT)
%           Sw    = wing surface area (FT^2)
%           lw    = location of wing leading edge-body intersection
%                   measured from the tip of the nose (FT)
%           bt    = tail span (FT)
%           St    = tail surface area (FT^2)
%           Mach  = Mach number
%           alpha = assumed angle of attack (DEG).  This is required for
%                   calculations in the non-linear angle of attack
%                   region.  If no value is entered for alpha,
%                   alpha = 0 is assumed.


if nargin < 10
      alpha = 0;
end

alpha = pi*alpha/180;

Sref = pi*(d/2)^2;

%  Calculate nose-body lift-curve-slope
CLaN = bdycla(d,l,ln,Mach,alpha);

%  Calculate the wing parameters
Aw = bw^2/Sw;
sw = 0.5*(d+bw);
Kbw = bigkbwa(Mach,Aw,d/2,sw);
Kwb = bigkwb(d/2,sw);
CLaW = clawsup(Mach,Aw);
crw = 2*Sw/bw;
SWPwing = atan(2*crw/bw);

%  Calculate the tail parameters
Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
At = bt^2/St;
st = 0.5*(d+bt);
Kbt = bigkbwna(Mt,At,d/2,st);
Ktb = bigkwb(d/2,st);
CLaT = clawsup(Mt,At);
crt = 2*St/bt;
lt = l-crt;
x0 = (lt-(lw+crw))/crw;
deda = Kwb*dedasup(Mach,SWPwing,x0);
CLa = CLaN+(Kbw+Kwb)*CLaW*Sw/Sref+(Kbt+Ktb)*CLaT*(1-deda)*nt*St/Sref;
```

CLAWBT

```
function cla = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha)
%  CLAWBT Returns the lift-curve-slope of a body-wing-tail combination.
%     The assumptions include triangular planform aerodynamic surfaces,
%     an ogival nose with a cylindrical afterbody.  The lift-curve-slope
%     is referenced to the area of the body diameter.
%
%           cla = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha)
%
%           d     = missile diameter (FT)
%           l     = missile length (FT)
%           ln    = length of nose (FT)
%           bw    = wing span (FT)
%           Sw    = wing surface area (FT^2)
%           lw    = location of wing leading edge-body intersection
%                   measured from the tip of the nose (FT)
%           bt    = tail span (FT)
%           St    = tail surface area (FT^2)
%           Mach  = Mach number
%           alpha = assumed angle of attack (DEG).  This is required for
%                   calculations in the non-linear angle of attack
%                   region.  If no value is entered for alpha,
%                   alpha = 0 is assumed.

if nargin < 10
     alpha = 0;
end

if Mach < 1
     cla = clamslsb(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha);
else
     cla = clamslsp(d,l,ln,bw,Sw,lw,bt,St,Mach,alpha);
end
```

CLAWSUB

```
function ClaW = clawsub(M,A,lc2,kappa)
%  CLAWSUB Calcuates the lift curve slope of a subsonic wing
%
%            ClaW = clawsub(M,A,lc2,kappa)
%
%            M      = Mach number
%            A      = aspect ratio
%            lc2    = sweep angle of half chord (RAD)
%            kappa  = wing efficiency factor

if M == 1
      M = 0.999;
end

b = sqrt(abs(1-M^2));

ClaW = 2*pi*A/(2+sqrt(A^2*b^2*(1+(tan(lc2))^2/b^2)/kappa^2+4));
```

CLAWSUP

```
function CLaW = clawsup(M,A)
%  CLAWSUP Returns the supersonic lift curve slope of a thin triangular
%     wing.
%
%           CLaW = clawsup(M,A)
%
%           CLaW  = lift curve slope
%           M     = Mach number
%           A     = aspect ratio

b = sqrt(M^2-1);

if b*A/4 >= 1           %  Supersonic leading edge
      CLaW = 4/b;
else                    %  Subsonic leading edge
      [K,E] = ellipke(sqrt(1-(b*A/4)^2));
      CLaW = pi*A/2/E;
end
```

CLDWBT

```
function CLd = cldwbt(d,bt,St,Mach)
%  CLDWBT Returns CLd, the coefficient of lift due to incidence angle
%     based on the base reference area.
%
%           CLd = cldwbt(d,bt,St,Mach)
%
%           d     = missile diameter
%           bt    = tail span
%           St    = tail size
%           Mach  = Mach number

Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
Sref = pi*d^2/4;
At = bt^2/St;
st = 0.5*(d+bt);

if Mach < 1

      lc2t = atan(2/At);
      kappat = 0.85;
      CLaT = clawsub(Mt,At,lc2t,kappat);

else

      CLaT = clawsup(Mt,At);

end

kbt = smlkbw(d/2,st);
ktb = smlkwb(d/2,st);

CLd = CLaT*nt*(kbt+ktb)*St/Sref;
```

CMADWBT

```
function cmad = cmadwbt(d,l,lcg,bw,Sw,lw,bt,St,Mach)
%  CMADWBT Calculates Cm(alpha dot) for a tail-controlled missile.
%
%           cmad = cmadwbt(d,l,lcg,bw,Sw,lw,bt,St,Mach)
%
%           d     = missile diameter
%           l     = missile length
%           lcg   = missile center of gravity measured from nose
%           bw    = wing span
%           Sw    = wing area
%           lw    = wing location measured from nose
%           bt    = tail span
%           St    = tail size
%           Mach  = Mach number

Sref = pi*(d/2)^2;
Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
At = bt^2/St;
st = 0.5*(d+bt);
crt = 2*St/bt;
lt = l-crt;

Aw = bw^2/Sw;
crw = 2*Sw/bw;

if Mach < 1

      lc2t = atan(2/At);
      kappat = 0.85;
      CLaT = clawsub(Mt,At,lc2t,kappat);

      Kbt = bigkbwsb(d/2,st);
      Ktb = bigkwb(d/2,st);

      xcrbt = xcrbwsub(Mt,At,d/2,st);
      xcrtb = xcrwbsub(Mt,At);

      lbt = lt+xcrbt*crt;
      ltb = lt+xcrtb*crt;

      xbt = lbt-lcg;
      xtb = ltb-lcg;

      xcrwb = xcrwbsub(Mach,Aw);

      xwt = ltb-(lw+xcrwb*crw);

      lH = l-St/bt-(lw+Sw/bw);
      lc4w = atan(3/Aw);
      deda = dedasub(Aw,bw,lc4w,0,lH,0);

      lc2w = atan(2/Aw);
      kappaw = 0.85;
      CLaW = clawsub(Mach,Aw,lc2w,kappaw);
      CLaW0 = clawsub(0,Aw,lc2w,kappaw);

      dedaM = deda*CLaW/CLaW0;
```

```
      cmad = -2*CLaT*(Kbt*xbt+Ktb*xtb)*nt*St*xwt*dedaM/Sref/d^2;

else

      Kbt = bigkbwna(Mt,At,d/2,st);
      Ktb = bigkwb(d/2,st);

      CLaT = clawsup(Mt,At);

      sw = 0.5*(d+bw);
      SWPwing = atan(2*crw/bw);
      Kwb = bigkwb(d/2,sw);
      x0 = (lt-(lw+crw))/crw;
      deda = Kwb*dedasup(Mach,SWPwing,x0);
      xcrwb = xcrwba(d/2,sw);

      betat = sqrt(Mt^2-1);

      xcrtb = xcrwba(d/2,st);

      if betat*At >= 0
            xcrbt = xcrbwnab(Mt,d/2,crt);
      else
            xcrbt = xcrbwabl(Mt,At,d/2,st);
      end

      lbt = lt+xcrbt*crt;
      ltb = lt+xcrtb*crt;

      xbt = lbt-lcg;
      xtb = ltb-lcg;

      xwt = ltb-(lw+xcrwb*crw);

      cmad = -2*CLaT*(Kbt*xbt+Ktb*xtb)*nt*St*xwt*deda/Sref/d^2;
end
```

CMASUB

```
function cma = cmasub(d,l,lcg,ln,bw,Sw,lw,bt,St,Mach,alpha)
%  CMASUB Returns the subsonic pitching-moment-curve slope for angle
%     of attack for a body-wing-tail combination.  The assumptions
%     include triangular planform aerodynamic surfaces, an ogival
%     nose with a cylindrical afterbody.  The pitching-moment-curve
%     slope is referenced to the area of the body diameter and is
%     valid only in the subsonic region ( < 0.95).
%
%           cma = cmasub(d,l,lcg,ln,bw,Sw,lw,bt,St,Mach,alpha)
%
%           d     = missile diameter (FT)
%           l     = missile length (FT)
%           lcg   = distance from nose to center of gravity (FT)
%           ln    = length of nose (FT)
%           bw    = wing span (FT)
%           Sw    = wing surface area (FT^2)
%           lw    = location of wing leading edge-body intersection
%                   measured from the tip of the nose (FT)
%           bt    = tail span (FT)
%           St    = tail surface area (FT^2)
%           Mach  = Mach number
%           alpha = assumed angle of attack (DEG).  This is required for
%                   calculations in the non-linear angle of attack
%                   region.  If no value is entered for alpha,
%                   alpha = 0 is assumed.

if nargin < 11
     alpha = 0;
end

alpha = alpha*pi/180;
Sref = pi*d^2/4;

%  Calculate nose parameters
CLaN = bdycla(d,l,ln,Mach,alpha);
cpn = cpogvsb(ln,d);
xn = lcg-cpn;

%  Calculate wing parameters
sw = 0.5*(d+bw);
Aw = bw^2/Sw;
crw = 2*Sw/bw;
kappaw = 0.85;
CLaW0 = clawsub(0,Aw,atan(2/Aw),kappaw);
CLaW = clawsub(Mach,Aw,atan(2/Aw),kappaw);
Kbw = bigkbwsb(d/2,sw);
Kwb = bigkwb(d/2,sw);
lc4w = atan(3/Aw);
xbw = lw+xcrbwsub(Mach,Aw,d/2,sw)*crw-lcg;
xwb = lw+xcrwbsub(Mach,Aw)*crw-lcg;

%  Calculate tail parameters
Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
st = 0.5*(d+bt);
At = bt^2/St;
crt = 2*St/bt;
kappat = 0.85;
```

```
Kbt = bigkbwsb(d/2,st);
Ktb = bigkwb(d/2,st);
lt = l-crt;
xbt = lt+xcrbwsub(Mt,At,d/2,st)*crt-lcg;
xtb = lt+xcrwbsub(Mt,At)*crt-lcg;
kappat = 0.85;
CLaT = clawsub(Mt,At,atan(2/At),kappat);
lH = l-St/bt-(lw+Sw/bw);
dedaM = dedasub(Aw,bw,lc4w,0,lH,0)*CLaW/CLaW0;

cmad = CLaN*xn-(Kbw*xbw+Kwb*xwb)*CLaW*Sw/Sref;
cmad = cmad - (Kbt*xbt+Ktb*xtb)*CLaT*nt*St*(1-dedaM)/Sref;
cma = cmad/d;
```

CMASUP

```
function cma = cmasup(d,l,lcg,ln,bw,Sw,lw,bt,St,Mach,alpha)
%  CMASUP Returns the supersonic pitching-moment-curve slope for angle
%     of attack for a body-wing-tail combination.  The assumptions
%     include triangular planform aerodynamic surfaces, an ogival
%     nose with a cylindrical afterbody.  The pitching-moment-curve
%     slope is referenced to the area of the body diameter and is
%     valid only in the supersonic region (1.2 - 5).
%
%           cma = cmasup(d,l,lcg,ln,bw,Sw,lw,bt,St,Mach,alpha)
%
%           d     = missile diameter (FT)
%           l     = missile length (FT)
%           lcg   = distance from nose to center of gravity (FT)
%           ln    = length of nose (FT)
%           bw    = wing span (FT)
%           Sw    = wing surface area (FT^2)
%           lw    = location of wing leading edge-body intersection
%                   measured from the tip of the nose (FT)
%           bt    = tail span (FT)
%           St    = tail surface area (FT^2)
%           Mach  = Mach number
%           alpha = assumed angle of attack (DEG).  This is required for
%                   calculations in the non-linear angle of attack
%                   region.  If no value is entered for alpha,
%                   alpha = 0 is assumed.

if nargin < 11
      alpha = 0;
end

alpha = alpha*pi/180;
Sref = pi*d^2/4;

%  Calculate nose parameters
CLaN = bdycla(d,l,ln,Mach,alpha);
cpn = cpogvemp(d,ln,Mach);
xn = lcg-cpn;

%  Calculate wing parameters
sw = 0.5*(d+bw);
Aw = bw^2/Sw;
crw = 2*Sw/bw;
betaw = sqrt(Mach^2-1);
Kbw = bigkbwa(Mach,Aw,d/2,sw);
Kwb = bigkwb(d/2,sw);
CLaW = clawsup(Mach,Aw);
SWPwing = atan(2*crw/bw);
xcrwb = xcrwba(d/2,sw);

if betaw*Aw >=0
      xcrbw = xcrbwabh(Mach,d/2,crw);
else
      xcrbw = xcrbwabl(Mach,Aw,d/2,sw);
end

xbw = lw+xcrbw*crw-lcg;
xwb = lw+xcrwb*crw-lcg;
```

```
%  Calculate tail parameters
Mt = 0.95*Mach;
betat = sqrt(Mt^2-1);
nt = (Mt/Mach)^2;
st = 0.5*(d+bt);
At = bt^2/St;
crt = 2*St/bt;
Kbt = bigkbwna(Mt,At,d/2,st);
Ktb = bigkwb(d/2,st);
lt = l-crt;
xcrtba = xcrwba(d/2,st);

if betat*At >= 0
      xcrbta = xcrbwnab(Mt,d/2,crt);
else
      xcrbta = xcrbwabl(Mt,At,d/2,st);
end

xbt = lt+xcrbta*crt-lcg;
xtb = lt+xcrtba*crt-lcg;

x0 = (lt-(lw+crw))/crw;
deda = Kwb*dedasup(Mach,SWPwing,x0);

CLaT = clawsup(Mt,At);

cmad = CLaN*xn-(Kbw*xbw+Kwb*xwb)*CLaW*Sw/Sref;
cmad = cmad - (Kbt*xbt+Ktb*xtb)*CLaT*nt*St*(1-deda)/Sref;
cma = cmad/d;
```

CMAWBT

```
function cma = cmawbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach,alpha)
%  CMAWBT Returns the pitching-moment-curve slope for angle of attack
%     of a body-wing-tail combination.  The assumptions include
%     triangular planform aerodynamic surfaces, an ogival nose with
%     a cylindrical afterbody.  The pitching-moment-curve slope is
%     referenced to the area of the body diameter.
%
%           cma = cmawbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach,alpha)
%
%           d     = missile diameter (FT)
%           l     = missile length (FT)
%           ln    = length of nose (FT)
%           lcg   = distance from nose to center of gravity (FT)
%           bw    = wing span (FT)
%           Sw    = wing surface area (FT^2)
%           lw    = location of wing leading edge-body intersection
%                   measured from the tip of the nose (FT)
%           bt    = tail span (FT)
%           St    = tail surface area (FT^2)
%           Mach  = Mach number
%           alpha = assumed angle of attack (DEG).  This is required for
%                   calculations in the non-linear angle of attack
%                   region.  If no value is entered for alpha,
%                   alpha = 0 is assumed.

if nargin < 11
      alpha = 0;
end

if Mach < 1
      cma = cmasub(d,l,lcg,ln,bw,Sw,lw,bt,St,Mach,alpha);
else
      cma = cmasup(d,l,lcg,ln,bw,Sw,lw,bt,St,Mach,alpha);
end
```

CMDSUB

```
function cmd = cmdsub(d,l,lcg,bt,St,Mach)
%  CMDSUB Returns the value of the pitching-moment-curve slope for tail
%     incidence angle for the complete wing-body-tail combination in
%     the subsonic region (<0.95).
%
%           cmd = cmdsub(d,l,lcg,bt,St,Mach)
%
%           d     = missile diameter
%           l     = missile length
%           lcg   = center of gravity measured from nose
%           bt    = tail span
%           St    = tail size
%           Mach  = Mach number

if Mach == 0
      Mach = 0.001;
end

Sref = pi*d^2/4;
Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
At = bt^2/St;
st = 0.5*(d+bt);
crt = 2*St/bt;
lt = l-crt;
kappat = 0.85;
lc2t = atan(2/At);

xcrbt = xcrbwsub(Mt,At,d/2,st);
xcrtb = xcrwbsub(Mt,At);

lbt = lt+xcrbt*crt;
ltb = lt+xcrtb*crt;

xbt = lbt-lcg;
xtb = ltb-lcg;

kbt = smlkbw(d/2,st);
ktb = smlkwb(d/2,st);

CLaT = clawsub(Mt,At,lc2t,kappat);

cmd = -(ktb*xtb+kbt*xbt)*CLaT*nt*St/Sref/d;
```

CMDSUP

```
function cmd = cmdsup(d,l,lcg,bt,St,Mach)
%  CMDSUP Returns the value of the pitching-moment-curve slope for tail
%      incidence angle for the complete wing-body-tail combination in
%      the supersonic region (1.2 - 5).
%
%            cmd = cmdsup(d,l,lcg,bt,St,Mach)
%
%            d     = missile diameter
%            l     = missile length
%            lcg   = center of gravity measured from nose
%            bt    = tail span
%            St    = tail size
%            Mach  = Mach number


Sref = pi*d^2/4;
Mt = 0.95*Mach;
betat = sqrt(Mt^2-1);
nt = (Mt/Mach)^2;
At = bt^2/St;
st = 0.5*(d+bt);
crt = 2*St/bt;
lt = l-crt;

if betat*At < 0
      xcrbtd = xcrbwabl(Mt,At,d/2,st);
else
      xcrbtd = xcrbwnab(Mt,d/2,crt);
end

xcrtbd = xcrwbd(d/2,st);

lbtd = lt+xcrbtd*crt;
ltbd = lt+xcrtbd*crt;

xbtd = lbtd-lcg;
xtbd = ltbd-lcg;

kbt = smlkbw(d/2,st);
ktb = smlkwb(d/2,st);

CLaT = clawsup(Mt,At);

cmd = -(ktb*xtbd+kbt*xbtd)*CLaT*nt*St/Sref/d;
```

CMDWBT

```
function cmd = cmdwbt(d,l,lcg,bt,St,Mach)
%  CMDWBT Returns the value of the pitching-moment-curve slope for tail
%     incidence angle for the complete wing-body-tail combination in
%     the subsonic and supersonic regions.  If the Mach number falls
%     between Mach 0.95 and 1.2, a message indicating transonic region
%     is returned.
%
%           cmd = cmdwbt(d,l,lcg,bt,St,Mach)
%
%           d     = missile diameter
%           l     = missile length
%           lcg   = center of gravity measured from nose
%           bt    = tail span
%           St    = tail size
%           Mach  = Mach number

if Mach <= 0.95
      cmd = cmdsub(d,l,lcg,bt,St,Mach);
elseif Mach >= 1.2
      cmd = cmdsup(d,l,lcg,bt,St,Mach);
else
      disp('Mach number inside transonic region.  Methodology invalid.')
end
```

CMQWBT

```
function cmq = cmqwbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach)
%  CMQWBT Calculates Cmq for a tail-controlled missile.
%
%            cmq = cmqwbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach)
%
%            d     = missile diameter
%            l     = missile length
%            ln    = nose length
%            lcg   = missile center of gravity measured from nose
%            bw    = wing span
%            Sw    = wing area
%            lw    = wing location measured from nose
%            bt    = tail span
%            St    = tail size
%            Mach  = Mach number

Sref = pi*(d/2)^2;
Mt = 0.95*Mach;
nt = (Mt/Mach)^2;
At = bt^2/St;
st = 0.5*(d+bt);
crt = 2*St/bt;
lt = l-crt;

Aw = bw^2/Sw;
crw = 2*Sw/bw;
sw = 0.5*(d+bw);

CLaN = bdycla(d,l,ln,Mach,0);

if Mach < 1

      cpn = cpogvsb(ln,d);
      xn = lcg-cpn;

      lc2t = atan(2/At);
      kappat = 0.85;
      CLaT = clawsub(Mt,At,lc2t,kappat);

      Kbt = bigkbwsb(d/2,st);
      Ktb = bigkwb(d/2,st);

      xcrbt = xcrbwsub(Mt,At,d/2,st);
      xcrtb = xcrwbsub(Mt,At);

      lbt = lt+xcrbt*crt;
      ltb = lt+xcrtb*crt;

      xbt = lbt-lcg;
      xtb = ltb-lcg;

      xcrbw = xcrbwsub(Mach,Aw,d/2,st);
      xcrwb = xcrwbsub(Mach,Aw);

      Kbw = bigkbwsb(d/2,sw);
      Kwb = bigkwb(d/2,sw);

      xcrbw = xcrbwsub(Mach,Aw,d/2,sw);
```

```
        xcrwb = xcrwbsub(Mach,Aw);

        lbw = lw+xcrbw*crw;
        lwb = lw+xcrwb*crw;

        xbw = lbw-lcg;
        xwb = lwb-lcg;

        lc2w = atan(2/Aw);
        kappaw = 0.85;
        CLaW = clawsub(Mach,Aw,lc2w,kappaw);

        cmq = (CLaN*xn^2+CLaT*(Kbt*xbt+Ktb*xtb)*nt*St*xtb/Sref);
        cmq = -2*(cmq+CLaW*(Kbw*xbw+Kwb*xwb)*Sw*xwb/Sref)/d^2;

else
        cpn = cpogvemp(d,ln,Mach);
        xn = lcg-cpn;

        Kbt = bigkbwna(Mt,At,d/2,st);
        Ktb = bigkwb(d/2,st);

        betat = sqrt(Mt^2-1);

        xcrtb = xcrwba(d/2,st);

        if betat*At >= 0
              xcrbt = xcrbwnab(Mt,d/2,crt);
        else
              xcrbt = xcrbwabl(Mt,At,d/2,st);
        end

        lbt = lt+xcrbt*crt;
        ltb = lt+xcrtb*crt;

        xbt = lbt-lcg;
        xtb = ltb-lcg;

        CLaT = clawsup(Mt,At);

        Kwb = bigkwb(d/2,sw);
        Kbw = bigkbwa(Mach,Aw,d/2,sw);

        xcrwb = xcrwba(d/2,sw);

        betaw = sqrt(Mach^2-1);

        if betaw*Aw >=0
              xcrbw = xcrbwabh(Mach,d/2,crw);
        else
              xcrbw = xcrbwabl(Mach,Aw,d/2,sw);
        end

        lbw = lw+xcrbw*crw;
        lwb = lw+xcrwb*crw;

        xbw = lbw-lcg;
        xwb = lwb-lcg;

        CLaW = clawsup(Mach,Aw);
```

```
      cmq = (CLaN*xn^2+CLaT*(Kbt*xbt+Ktb*xtb)*nt*St*xtb/Sref);
      cmq = -2*(cmq+CLaW*(Kbw*xbw+Kwb*xwb)*Sw*xwb/Sref)/d^2;
end
```

CNTRPRSS

```
function cntrprss()
%  CNTRPRSS  Summarizes the applicaable center of pressure m-files
%     available.
%
%     Subsonic conditions:
%
%           (x/cr)w(b)alpha or delta = xcrwbsub
%           (x/cr)b(w)alpha or delta = xcrbwsub
%
%     Supersonic conditions:
%
%           (x/cr)w(b)alpha = xcrwba
%           (x/cr)w(b)delta = xcrwbd
%
%           (x/cr)b(w)alpha or delta
%
%                 b*A < 0                  = xcrbwabl
%
%                 b*A >= 0
%                       afterbody    = xcrbwabh
%                       no afterbody       = xcrbwnab
```

CPOGVEMP

```
function cpn = cpogvemp(d,ln,M)
%  CPOGVEMP Calculates the center of pressure of an ogival nose measured
%     from the tip.
%
%           cpn = cpogvemp(d,ln,M)
%
%           cpn    = center of pressure of the nose
%           d      = diameter
%           ln     = nose length
%           M      = Mach number

sigma = 2*atan(d/2/ln)*180/pi;

P = (0.083+0.096/M^2)*(sigma/10)^1.69;

cpn = 0.5*(50*(M+18)+7*M^2*P*(5*M-18))*ln/(40*(M+18)+7*M^2*P*(4*M-3));
```

CPOGVSB

```
function xcp = cpogvsb(ln,d)
%  CPOGVSB Calculates the center of pressure for a tangent ogive using
%     slender body theory.
%
%           xcp = cpogvsb(ln,d)
%
%           xcp    = center of pressure measured from the tip of the nose
%           ln     = length of nose from the tip to the base
%           d      = radius of the base
%
%     The lower value of the fineness ratio is dependent on Mach number
%     and does not apply to blunt ogive such as a hemisphere.

%     David A. Ekker
%     Developed from NACA 1307 slender body theory

Vn = vologv1(ln,d);

xcp = ln-4*Vn/pi/d^2;
```

CYLIYY

```
function iyy = cyliyy(mass,radius,height)
%  CYLIYY Calculates the Iyy for a circular cylinder
%           iyy = cyliyy(mass,radius,height)

iyy = mass.*(3*radius.^2+height.^2)/12;
```

DEDASUB

```
function deda = dedasub(A,b,lc4,lambda,lH,hH)
%  DEDASUB Calculates de/da for subsonic conditions using an empirical
%     method.
%
%           deda = dedasub(A,b,lc4,lambda,lH,hH)
%
%           A      = aspect ratio
%           b      = span
%           lc4    = leading edge quarter chord sweep angle (RAD)
%           lambda= taper ratio
%           lH     = distance between the wing MAC quarter chord point
%                    and the quarter-chord point of the tail
%           hH     = distance between the qurter chord point of the tail
%                    MAC relative to the plane of the wing root chord
%
%     From Sanders, K. L., "An Empirical Method for the Estimation of
%     Downwash",
%     Ryan Rpt. No. 29254-2A, 1967.


KA = 1/A-1/(1+A^1.7);

Kl = (10-3*lambda)/7;

KH = (1-abs(hH/b))/(2*lH/b)^(1/3);

deda = 4.44*(KA*Kl*KH*(cos(lc4))^0.5)^1.19;
```

DEDASUP

```
function dedaw = dedasup(M,LAMBDA,x0)
%  DEDALPHW Calculates (de/da)w vs downstream position of delta wings in
%     supersonic flow with in-the-plane symmetry.
%
%           dedaw = dedasup(M,LAMBDA,x0)
%
%           M      = Mach Number
%           LAMBDA= wing leading edge sweep angle (rad)
%           x0     = distance aft of wing trailing edge to tail leading
%                    edge divided by wing root chord
%
%     The resulting value for (deda)w needs to be multiplied by the
%     coefficient of lift of the wing in the presence of the body, Kw(b)

psi = pi/2 - LAMBDA;

t0 = tan(psi)*sqrt(M^2-1);

if x0 < 1.0
      e2 = 0.81;
elseif x0 <= 1.3
      p21=[-1.66666666666488e+000; 5.999999999999325e+000; -
6.783333333332485e+000;
            3.259999999999646e+000];
      e2 = polyval(p21,x0);
elseif x0 < 2.3
      p22=[-1.279239755663028e+002; 1.843272381677282e+003;
           -1.155286798315866e+004; 4.113252460613160e+004;
           -9.098009805564612e+004; 1.280057606805492e+005;
           -1.118657697776747e+005; 5.551390292092566e+004;
           -1.197584408390502e+004];
      e2 = polyval(p22,x0);
else
      e2 = 0.97;
end

if x0 < 1.0
      e4 = 0.65;
elseif x0 <= 1.6
      p41=[-4.166666666665245e+000; 2.083333333332625e+001;
           -3.845833333332031e+001; 3.134166666665624e+001;
           -8.899999999996956e+000];
      e4 = polyval(p41,x0);
elseif x0 <= 3.0
      p42=[ 4.475474425067136e+000;-8.198016093532951e+001;
            6.529374943388477e+002;-2.952920552556513e+003;
            8.292996506025491e+003;-1.480813974021849e+004;
            1.641631929076350e+004;-1.032958858524409e+004;
            2.825103492654550e+003];
      e4 = polyval(p42,x0);
else
      e4 = 0.87;
end

if x0 < 1.0
      e6 = 0.54;
elseif x0 <= 1.4
```

```
      p61=[ 1.600881353985564e-014;-6.387361790702744e-014;
            1.000000000000832e-001;4.399999999999644e-001];
      e6 = polyval(p61,x0);
elseif x0 <= 1.8
      p62=[-3.333333333309070e+001; 2.116666666651215e+002;
           -5.026666666629868e+002;5.294833333294496e+002;
           -2.082299999984673e+002];
      e6 = polyval(p62,x0);
elseif x0 < 3.6
      p63=[ 6.356470889426016e-003;-5.240723295023863e-002;
           -2.580914111953288e-002; 1.556718856871693e+000;
           -6.218769329470563e+000; 1.005756492417748e+001;
           -5.269710040962651e+000];
      e6 = polyval(p63,x0);
else
      e6 = 0.79;
end


if x0 < 1.0
      e8 = 0.44;
elseif x0 <= 1.4
      p81=[-1.666666666665405e+001; 7.999999999993737e+001;
           -1.433333333332168e+002; 1.136499999999038e+002;
           -3.320999999997028e+001];
      e8 = polyval(p81,x0);
elseif x0 <= 2.2
      p82=[-8.680554787360018e+002; 1.252976079851500e+004;
           -7.886457638672026e+004; 2.827083084908162e+005;
           -6.312798925797358e+005; 8.991348588842052e+005;
           -7.977027436345048e+005; 4.030393822169418e+005;
           -8.878791230326184e+004];
      e8 = polyval(p82,x0);
elseif x0 < 3.4
      p83=[3.866380123250028e+000;-9.169392757855618e+001;
           9.469642944017384e+002;-5.562064279886374e+003;
           2.032063141263390e+004;-4.728372988378842e+004;
           6.842861819191089e+004;-5.630916695234881e+004;
           2.017213892626900e+004];
      e8 = polyval(p83,x0);
else
      e8 = 0.7;
end

if t0 <= 0.2
      dedaw = (e2-e4)*(0.2-t0)/0.2 + e2;
elseif t0 <= 0.4
      dedaw = (e2-e4)*(0.4-t0)/0.2 + e4;
elseif t0 <= 0.6
      dedaw = (e4-e6)*(0.6-t0)/0.2 + e6;
elseif t0 <= 0.8
      dedaw = (e6-e8)*(0.8-t0)/0.2 + e8;
elseif t0 > 0.8
      dedaw = e8 - (e6-e8)*(t0-0.8)/0.2;
end

if dedaw < 0
      dedaw = 0;
end
```

DRAGEX1

```
%********************* DRAGEX1 *********************
%
%  DRAGEX1 generates a drag versus Mach number plot
%      for a missile using CDOWBT


%  Missile data

d = 0.6;
l = 12;
ln = 2*d;
bw = 2.5817;
Sw = 2.3798;
thickw = 0.04;
bt = 1.2950;
St = 0.5988;
thickt = 0.04;
alt = 10000;
M = linspace(0,4);

Sref = pi*d^2/4;
drag = zeros(size(M));


for k = 1:length(M)
      cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M(k),alt);
      drag(k) = dynmprss(M(k),alt)*cdomsl*Sref;
end

plot(M,drag)
title('Zero Lift Drag Versus Mach Number')
xlabel('Mach Number')
ylabel('Drag (lb)')
```

DRAGFCTR

```
function n=dragfctr(fineness)
%  DRAGFCTR Returns the drag proportionality factor, n.
%
%           n = dragfctr(fineness)
%
%           fineness = body fineness ratio


p = [-3.610649936485454e-12;8.309751611041349e-10;-7.347789500313429e-8;
      3.037472628776377e-6;-5.190776944854130e-5;-1.773312264112830e-4;
      2.135202437974324e-2;  5.187105185342232e-1];

n = polyval(p,fineness);
```

DYNMPRSS

```
function qbar = dynmprss(Mach,altitude,iunits)
%  DYNMPRSS Calculates the dynamic pressure, qbar, for the input Mach
%     number and altitude in feet.  Atmospheric properties are based on
%     a 1962 Standard Atmosphere.
%
%           qbar = dynmprss(Mach,altitude,iunits)
%
%     English is the default unit type.  If Metric units are required,
%     use:
%           qbar = dynmprss(Mach,altitude,1)

if nargin < 3
      qbar = 0.5*atmsdens(altitude)*Mach^2*atmssofs(altitude)^2;
else
      qbar = 0.5*atmsdens(altitude,1)*Mach^2*atmssofs(altitude,1)^2;
end
```

K2MK1

```
function amf = k2mk1(fineness)
%  K2MK1 Calculates the apparent mass factor for a given fineness ratio.
%
%           amf = k2mk1(fineness)

%     Written by David A. Ekker

pa = [-6.3424e-7 3.1805e-5 -6.0351e-4 5.0080e-3 -9.9444e-3 -1.1439e-1
7.7981e-1 -6.6077e-1];
pb = [1.6667e-3 9.4667e-1];

[m,n] = size(fineness);

amf = zeros(m,n);

l = m*n;

for k = 1:l

      if fineness(k) < 0
            fineness(k) = 0;
      end

      if fineness(k) < 14
            amf(k) = polyval(pa,fineness(k));
      else
            amf(k) = polyval(pb,fineness(k));
      end

      if amf(k) < 0
            amf(k) = 0;
      end

end
```

MSLDSPLY

```
function
msldsply(d,l,ln,lw,crw,bw,lambdaw,SWPwing,lt,crt,bt,lambdat,SWPtail)
%  MSLDSPLY Plots the outline of a missile
%
%   msldsply(d,l,ln,lw,crw,bw,lambdaw,SWPwing,lt,crt,bt,lambdat,SWPtail)
%
%           d       = missile diameter
%           l       = missile length
%           ln      = nose length
%           lw      = location of leading edge of wing at wing-body
%                     juncture
%           crw     = wing root chord
%           bw      = wing exposed span
%           lambdaw= wing taper ratio
%           SWPwing= wing leading edge sweep angle (DEG)
%           lt      = location of leading edge of tail at tail-body
%                     juncture
%           crt     = tail root chord
%           bt      = tail exposed span
%           lambdat= tail taper ratio
%           SWPtail= tail leading edge sweep angle (DEG)

%  Nose
R = d/4+ln^2/d;
a = R-d/2;
xn = linspace(0,ln,100);
x = ln-xn;
yn = (R^2-x.^2).^0.5-a;
%  Body
xb = [xn ln l l];
yb = [yn d/2 d/2 0];
%  Wing
ctw = crw*lambdaw;
lwtl = lw+bw*tan(SWPwing*pi/180)/2;
lwtt = lwtl+ctw;
lwrt = lw+crw;
xw = [lw lwtl lwtt lwrt];
yw = [d/2 (d/2+bw/2) (d/2+bw/2) d/2];
wx = [lw lwrt];
wy = [0 0];
%  Tail
ctt = crt*lambdat;
lttl = lt+bt*tan(SWPtail*pi/180)/2;
lttt = lttl+ctt;
ltrt = lt+crt;
xt = [lt lttl lttt ltrt];
yt = [d/2 (d/2+bt/2) (d/2+bt/2) d/2];
tx = [lt ltrt];
ty = [0 0];
xa = 1.1*l;
ya = 0;
%  Make the body blue
b = ['-b'];
%  Make the fins green
s = ['-g'];
plot(xb,yb,b,xb,-yb,b,xw,yw,s,xw,-yw,s,wx,wy,s,xt,yt,s,xt,-
yt,s,tx,ty,s,xa,ya,s)
axis('equal')
```

MSLIYY1

```
function Iyy = msliyy1(d,gclen,gcwt,whlen,whwt,plen,pwt)
%  MLSIYY1 Calculates the moment of inertia Iyy of a missile about the
%     Y axis at the center of gravity assuming a guidance/control-
%     warhead-propulsion subsection arrangement.  The center of
%     gravity of the component is one-half the length of the
%     component and the subsection is modeled by a uniform circular
%     cylinder.
%
%           Iyy = msliyy1(d,gclen,gcwt,whlen,whwt,plen,pwt)
%
%           d     = missile diameter
%           gclen = guidance/control subsection length
%           gcwt  = guidance/control subsection weight
%           whlen = warhead subsection weight
%           whwt  = warhead subsection weight
%           plen  = propulsion subsection length
%           pwt   = propulsion subsection weight


%  Find the center of gravity
xcgcomp = [gclen/2 (gclen+whlen/2) (gclen+whlen+plen/2)];
weight = [gcwt whwt pwt];

xcg = xcgfnd1(xcgcomp,weight);

g = 32.174;
r = d/2;
Iyy0 = [cyliyy(gcwt/g,r,gclen) cyliyy(whwt/g,r,whlen)
cyliyy(pwt/g,r,plen)];

Iyy = sum(Iyy0)+sum((weight/g)*(xcgcomp'-xcg).^2);
```

MSLREX1

```
%********************MSLREX1********************
%
%  MSLREX1 is used as an example of plotting a missile's
%     response to a step tail deflection

d = 0.6;
l = 12;
ln = 2*d;
W = 339;
lcg = 6.5553;
Iy = 103.96;
bw = 2.5817;
Sw = 2.3798;
lw = 5.5767;
bt = 1.295;
St = 0.5988;
Mach = 2.5;
alt = 10000;

[wn,z,p,af,tp,Mp] = mslrspns(d,l,ln,W,lcg,Iy,bw,Sw,lw,bt,St,Mach,alt)
```

MSLRSPNS

```
function [wn,z,p,af,tp,Mp] =
mslrspns(d,l,ln,W,lcg,Iy,bw,Sw,lw,bt,St,Mach,alt,d0,dsplyoff)
%  MSLRSPNS Calculates the natural frequency and damping factor for a
%     tail-controlled missile.
%
%     [wn,z,p,af,tp,Mp] =
%          mslrspns(d,l,ln,W,lcg,Iy,bw,Sw,lw,bt,St,Mach,alt,d0,dsplyoff)
%
%     wn     = natural frequency
%     z      = damping factor
%     p      = system poles
%     af     = final angle of attack (DEG)
%     tp     = peak time
%     Mp     = maximum overshoot
%     d      = missile diameter
%     l      = missile length
%     ln     = nose length
%     W      = missile weight
%     lcg    = missile center of gravity measured from nose
%     Iy     = moment of inertia about the y axis
%     bw     = wing span
%     Sw     = wing area
%     lw     = wing location measured from nose
%     bt     = tail span
%     St     = tail size
%     Mach   = Mach number
%     alt    = altitude
%     d0     = tail deflection angle (DEG).  d0 = -15 degrees if no
%              value is entered
%     dsplyoff= any value entered will prevent a plot of the response
%              from being displayed

if nargin < 14
     d0 = -15*pi/180;
else
     d0 = d0*pi/180;
end

qbar = dynmprss(Mach,alt);
Sref = pi*d^2/4;
g = 32.174;
a = atmssofs(alt);
m = W/g;
V = Mach*a;

Cmq = cmqwbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach);
CLa = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach);
Cma = cmawbt(d,l,ln,lcg,bw,Sw,lw,bt,St,Mach);
Cmad = cmadwbt(d,l,lcg,bw,Sw,lw,bt,St,Mach);

wn = (-qbar*Sref*d*(qbar*Sref*(d/2/V)*Cmq*CLa/m/V+Cma)/Iy)^0.5;

z = (qbar*Sref*CLa/m/V-qbar*Sref*d*(d/V/2)*(Cmq+Cmad)/Iy)/2/wn;

p = roots([1 2*z*wn wn^2]);

Cmd = cmdwbt(d,l,lcg,bt,St,Mach);
CLd = cldwbt(d,bt,St,Mach);
```

```
K = qbar*Sref*d*(Cmd+qbar*Sref*CLd/m/V)/Iy;

af = d0*K/wn^2;

t = 0:0.025:10;
t = t/wn;

[r,tp,Mp] = stprspns(t,wn,z);

at = af*r;
af = af*180/pi;

if nargin < 15
      clf
      plot(t,at*180/pi)
      xlabel('Time (sec)')
      ylabel('Angle of Attack (deg)')
      title('Missile Response to a Step Input')
end
```

PARAPLT1

```
function paraplt1(a,b,distort,plotsize)
%  PARAPLT1 Creates a parametric plot of input vector 'a' and output
%     matrix 'b'.
%
%     paraplt1(a,b,distort,plotsize)
%
%     'a' must be a 1 x n matrix.  'b' must be a m x n matrix, where m
%     is at least 3.  Try using paraplt1(a,b) first.  Default value of
%     distort and plotsize is 1.
%
%     If the plot lines are too close together, pick a value of
%     distort > 1.
%     (Note: Distort must be > 0.)  To increase the blank area around
%     the outside of the plot, pick a value of plotsize > 1.
%     (Note: try +.1 increments)

%     David A. Ekker

if nargin < 4, plotsize = 1; end
if nargin < 3, distort=1; end

[m n]=size(b);

x=zeros(m,n);

for i=1:m,
      x(i,:)=a+(i-1)*distort;
end

t=linspace(1,n,m);

u=zeros(m,m);
v=zeros(m,m);

for i=1:m,
      u(:,i)=x(:,t(i));
      v(:,i)=b(:,t(i));
end

%  Fix the plot axis
jmax=max(max(b));
jmin=min(min(b));
imax=max(max(x));
imin=min(min(x));

vs=plotsize*(jmax-jmin)/10;
hs=plotsize*(imax-imin)/10;
jmax=jmax+vs;
jmin=jmin-vs;
imax=imax+hs;
imin=imin-hs;
clf
plot(x',b',u,v)
axis([imin imax jmin jmax])
%  Turn off the individual tick mark labels.  The numbers don't mean
anything.
set(gca,'XTickLabels',['';'';'';'';'';'';'';'';'';'';'';'';'';'';''])
set(gca,'YTickLabels',['';'';'';'';'';'';'';'';'';'';'';'';'';'';''])
```

PCHAR1

```
function [pwt,plen] = pchar1(d,WL,dVb,Isp,rhop,zetam,vp,nzl)
%  PCHAR1 generates the weight and length of the booster assuming the
%     drag is zero during the boost phase.
%
%           [pwt,plen] = pchar1(d,WL,dVb,T,D,Isp,rhop,zetam,vp,nzl)
%
%           pwt   = propulsion subsection weight (lb)
%           plen  = propulsion subsection length (ft)
%           WL    = missile launch weight (lb)
%           dVb   = incremental velocity due to boost (ft/sec)
%           Isp   = specific impulse (sec)
%           rhop  = density of the propellant (LB/FT^3)
%           zetam = rocket motor mass ratio, assumed to be = 0.9
%                   unless a value ~=0 is entered.
%           vp    = volumetric packing factor, assumed to be = 0.85
%                   unless a value ~=0 is entered.
%           nzl   = nozzle length as a fraction of the length of
%                   the propellant.  A value of 0.3 is assumed unless
%                   another value is entered.

if nargin < 6
      zetam = 0.9;
      vp    = 0.85;
      nzl = .3;
elseif nargin < 7
      vp = 0.85;
      nzl = .3;
      if zetam == 0
            zetam = 0.9;
      end
elseif nargin < 8
      if zetam == 0
            zetam = 0.9;
      end
      if vp == 0
            vp = 0.85;
      end
      nzl = .3;
elseif nargin < 9
      if zetam == 0
            zetam = 0.9;
      end
      if vp == 0
            vp = 0.85;
      end
end


g = 32.174;

Wp = WL*(1-exp(-dVb/g/Isp));
pwt = Wp/zetam;

plen = (1+nzl)*4*Wp/vp/pi/d^2/rhop;
```

PCHAR2

```
function [pwt,plen] = pchar2(d,WL,dVb,T,D,Isp,rhop,zetam,vp,nzl)
%  PCHAR2 generates the weight and length of the booster assuming the
%     drag is an average value, D, during the boost phase.
%
%           [pwt,plen] = pchar2(d,WL,dVb,T,D,Isp,rhop,zetam,vp,nzl)
%
%           pwt    = propulsion subsection weight (lb)
%           plen   = propulsion subsection length (ft)
%           WL     = missile launch weight (lb)
%           dVb    = incremental velocity due to boost (ft/sec)
%           T      = booster thrust (lb)
%           D      = average drag (lb)
%           Isp    = specific impulse (sec)
%           rhop   = density of the propellant (LB/FT^3)
%           zetam  = rocket motor mass ratio, assumed to be = 0.9
%                    unless a value ~=0 is entered.
%           vp     = volumetric packing factor, assumed to be = 0.85
%                    unless a value ~=0 is entered.
%           nzl    = nozzle length as a fraction of the length of
%                    the propellant.  A value of 0.3 is assumed unless
%                    another value is entered.

if nargin < 8
      zetam = 0.9;
      vp    = 0.85;
      nzl = .3;
elseif nargin < 9
      vp = 0.85;
      nzl = .3;
      if zetam == 0
            zetam = 0.9;
      end
elseif nargin < 10
      if zetam == 0
            zetam = 0.9;
      end
      if vp == 0
            vp = 0.85;
      end
      nzl = .3;
elseif nargin < 11
      if zetam == 0
            zetam = 0.9;
      end
      if vp == 0
            vp = 0.85;
      end
end


g = 32.174;

Wp = WL*(1-exp(-dVb/g/(1-D/T)/Isp));
pwt = Wp/zetam;

plen = (1+nzl)*4*Wp/vp/pi/d^2/rhop;
```

PROPEX1

```
%*********************PROPEX1*********************
%
%  PROPEX1 caculates propulsion subsection weights
%      and lengths for a missile.  The first
%      attempt assumes drag = 0,  In the second
%      case, the drag is calculated and the
%      average value is used.

%  Missile data

d = 0.6;
l = 12;
ln = 2*d;
bw = 2.5817;
Sw = 2.3798;
thickw = 0.04;
bt = 1.2950;
St = 0.5988;
thickt = 0.04;
alt = 10000;
M = linspace(0,4);
WL = 339;
Isp = 260;
rho = 104;
dVb = 4*atmssofs(alt);

Sref = pi*d^2/4;
drag = zeros(size(M));


for k = 1:length(M)
      cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M(k),alt);
      drag(k) = dynmprss(M(k),alt)*cdomsl*Sref;
end

davg = mean(drag);

disp('Assume: boost to Mach =4, drag = 0, Isp = 260 sec and rho = 104
lb/ft^3')
[pwt,plen] = pchar1(d,WL,dVb,Isp,rho)

disp('Assume: drag = average drag, T = 10000 and other values the
same.')
T =10000;

[pwt,plen] = pchar2(d,WL,dVb,T,davg,Isp,rho)
```

PROPEX2

```
function propex2(Nb,Ng)
%  PROPEX2 Generates the velocity profile and range plot for a boost-
%     glide missile trajectory.
%
%           propex2(Nb,Ng)
%
%           Nb     = Number of iterations used for the boost phase
%           Ng     = Number of iterations used during the glide
%                    phase.  100 is the default value for the number
%                    of iterations.

tic

if nargin < 1
      Nb = 100;
      Ng = 100;
elseif nargin < 2
      Ng = Nb;
end

%  Missile aerodynamic data

d = 0.6;                  % Missile diameter
l = 12;                   % Missile Length
ln = 2*d;                 % Nose length
bw = 2.5817;              % Wing Span
Sw = 2.3798;              % Wing area
thickw = 0.04;            % Wing thickness
bt = 1.2950;              % Tail Span
St = 0.5988;              % Tail area
thickt = 0.04;            % Tail thickness
alt = 10000;              % Altitude
Sref = pi*d^2/4;

%  Gravitational constant
g = 32.174;

%  Propulsion data booster

WL = 339;    % Launch weight
WP = 140;    % Propellant weight
tb = 3.65;   % Booster burn time
T = 10000;   % Booster thrust
v0 = 400;    % Launch velocity
tg = 120;    % Glide time (sec)


%************************************************************************
%*****USER DOES NOT NEED TO ENTER ANY FURTHER DATA**********************
%
%
%     The algorithm used assumes (1) the acceleration versus time curve
% is linear over dt (2) the time interval used throughout the procedure
% is a constant value.  The algorithm can be found in Chin for the
% trajectory example and in most publications on numerical methods.
%
%                  V = V(n-1)+(3*a(n-1)-a(n-2))*dt/2
%
```

```
%                  D = D(n-1)+(3*V(n-1)-V(n-2))*dt/2
%

%  Initialize boost parameters
sofs = atmssofs(alt);
rho  = atmsdens(alt);

t     = linspace(0,tb,Nb);          % Time of boost
dt    = t(2)-t(1);                  % Time interval
a     = zeros(size(t));       % Acceleration
v     = zeros(size(t));       % Velocity
dist  = zeros(size(t));       % Range

W   = WL-WP*t/tb;             % Weight of the missile during boost
Wg0 = W(Nb);

% Calculate the drag at the initial velocity
M     = v0/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt);
drag  = rho*v0^2*cdomsl*Sref/2;

% Calculate the initial acceleration and subsequent velocity change
a(1)   = (T-drag)*g/W(1);
v(1)   = a(1)*dt+v0;
vavg   = (v0+v(1))/2;
dist(1)= vavg*dt;

M      = v(1)/sofs;
cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt);
drag   = rho*v(1)^2*cdomsl*Sref/2;
a(2)   = (T-drag)*g/W(2);
v(2)   = (a(1)+a(2))*dt/2+v(1);
dist(2)= dist(1)+(3*v(1)-v0)*dt/2;

for k = 3:length(t)
      M      = v(k-1)/sofs;
      cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt);
      drag   = rho*v(k-1)^2*cdomsl*Sref/2;
      a(k)   = (T-drag)*g/W(k);
      v(k)   = (3*a(k-1)-a(k-2))*dt/2+v(k-1);
      dist(k)= dist(k-1)+(3*v(k-1)-v(k-2))*dt/2;
end

disp('Completed boost phase, calculating glide phase')

% Initialize glide parameters
tg    = linspace(tb,tg,Ng);
dtg   = tg(2)-tg(1);
ag    = zeros(size(tg));
vg    = zeros(size(tg));
distg = zeros(size(tg));


M       = v(Nb)/sofs;
cdomsl  = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,0,1);
drag    = rho*v(Nb)^2*cdomsl*Sref/2;
ag(1)   = -drag*g/Wg0;
vg(1)   = ag(1)*dtg+v(Nb);
vavg    = (v(Nb)+vg(1))/2;
distg(1)= vavg*dtg+dist(Nb);
```

```
M       = vg(1)/sofs;
cdomsl  = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,0,1);
drag    = rho*vg(1)^2*cdomsl*Sref/2;
ag(2)   = -drag*g/Wg0;
vg(2)   = (ag(1)+ag(2))*dtg/2+vg(1);
distg(2)= distg(1)+(3*vg(1)-v(Nb))*dtg/2;

for kg = 3:length(tg)
      M         = vg(kg-1)/sofs;
      cdomsl    = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,0,1);
      drag      = rho*vg(kg-1)^2*cdomsl*Sref/2;
      ag(kg)    = -drag*g/Wg0;
      vg(kg)    = (3*ag(kg-1)-ag(kg-2))*dtg/2+vg(kg-1);
      distg(kg)= distg(kg-1)+(3*vg(kg-1)-vg(kg-2))*dtg/2;
end

toc

figure(1)
clf
plot(t,v,tg,vg)
title('Velocity Versus Time - PROPEX2')
xlabel('Time (sec)')
ylabel('Velocity (ft/sec)')

figure(2)
clf
plot([t,tg],[dist/6076.11549,distg/6076.11549])
title('Range Versus Time - PROPEX2')
xlabel('Time (sec)')
ylabel('Range (NM)')

figure(3)
clf
plot([t,tg],[a,ag]/g)
title('Acceleration Versus Time - PROPEX2')
xlabel('Time (sec)')
ylabel('Acceleration (g)')
```

PROPEX3

```
function propex3(Nb,Ng)
%  PROPEX3 Generates the velocity profile and range plot for a boost-
%     glide missile trajectory using a third degree multistep method for
%     integration.
%
%           propex3(Nb,Ng)
%
%           Nb    = Number of iterations used for the boost phase
%           Ng    = Number of iterations used during the glide
%                   phase.  100 is the default value for the number
%                   of iterations.

tic

if nargin < 1
      Nb = 100;
      Ng = 100;
elseif nargin < 2
      Ng = Nb;
end

%  Missile aerodynamic data

d = 0.6;                  % Missile diameter
l = 12;                   % Missile Length
ln = 2*d;                 % Nose length
bw = 2.5817;              % Wing Span
Sw = 2.3798;              % Wing area
thickw = 0.04;            % Wing thickness
bt = 1.2950;              % Tail Span
St = 0.5988;              % Tail area
thickt = 0.04;            % Tail thickness
alt = 10000;              % Altitude
Sref = pi*d^2/4;
nf = 1.1;                 % Friction Factor

%  Gravitational constant
g = 32.174;

%  Propulsion data booster

WL = 339;    % Launch weight
WP = 140;    % Propellant weight
tb = 3.65;   % Booster burn time
T = 10000;   % Booster thrust
v0 = 400;    % Launch velocity
tg = 120;    % Glide time (sec)

%  Initialize boost parameters

t     = linspace(0,tb,Nb);     % Time of boost
dt    = t(2)-t(1);             % Time interval
a     = zeros(size(t));        % Acceleration
v     = zeros(size(t));        % Velocity
dist  = zeros(size(t));        % Range
sofs  = atmssofs(alt);         % Speed of sound
rho   = atmsdens(alt);         % Density
```

```
W = WL-WP*t/tb;                    % Weight of the missile during boost
Wg0 = W(Nb);

% Calculate the first two increments using fourth-order Runga-Kutta
single-step method
M      = v0/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*v0^2*cdomsl*Sref/2;
k1     = dt*(T-drag)*g/W(1);
vk2    = (v0+k1/2);
M      = vk2/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*vk2^2*cdomsl*Sref/2;
tk2    = t(1)+dt/2;
Wk2    = WL-WP*tk2/tb;
k2     = dt*(T-drag)*g/Wk2;
vk3    = (v0+k2/2);
M      = vk3/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*vk3^2*cdomsl*Sref/2;
k3     = dt*(T-drag)*g/Wk2;
vk4    = v0+k3;
M      = vk4/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*vk3^2*cdomsl*Sref/2;
k4     = dt*(T-drag)*g/W(2);
a(1)   = k1/dt;
v(1)   = v0+(k1+2*k2+2*k3+k4)/6;
vavg   = (v0+v(1))/2;
dist(1) = vavg*dt;

M      = v(1)/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*v(1)^2*cdomsl*Sref/2;
k1     = dt*(T-drag)*g/W(2);
vk2    = (v(1)+k1/2);
M      = vk2/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*vk2^2*cdomsl*Sref/2;
tk2    = t(2)+dt/2;
Wk2    = WL-WP*tk2/tb;
k2     = dt*(T-drag)*g/Wk2;
vk3    = (v(1)+k2/2);
M      = vk3/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*vk3^2*cdomsl*Sref/2;
k3     = dt*(T-drag)*g/Wk2;
vk4    = v(1)+k3;
M      = vk4/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag   = rho*vk3^2*cdomsl*Sref/2;
k4     = dt*(T-drag)*g/W(3);
a(2)   = k1/dt;
v(2)   = v(1)+(k1+2*k2+2*k3+k4)/6;
dist(2) = dist(1)+(3*v(1)-v0)*dt/2;

M       = v(3)/atmssofs(alt);
cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
drag    = rho*v(3)^2*cdomsl*Sref/2;
a(3)    = (T-drag)*g/W(3);
v(3)    = (3*a(2)-a(1))*dt/2+v(2);
```

```
dist(3)= dist(2)+(23*v(2)-16*v(1)+5*v0)*dt/12;

for k = 4:length(t)
      M      = v(k-1)/sofs;
      cdomsl = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf);
      drag   = rho*v(k-1)^2*cdomsl*Sref/2;
      a(k)   = (T-drag)*g/W(k);
      v(k)   = v(k-1)+(23*a(k-1)-16*a(k-2)+5*a(k-3))*dt/12;
      dist(k)= dist(k-1)+(23*v(k-1)-16*v(k-2)+5*v(k-3))*dt/12;
end

disp('Completed boost phase, calculating glide phase')

% Initialize glide parameters
tg    = linspace(tb,tg,Ng);
dtg   = tg(2)-tg(1);
ag    = zeros(size(tg));
vg    = zeros(size(tg));
distg = zeros(size(tg));
T     = 0;

% Calculate the first two increments using fourth-order Runga-Kutta
single-step method
M     = v(Nb)/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*v(Nb)^2*cdomsl*Sref/2;
k1    = dtg*(T-drag)*g/Wg0;
vk2   = (v(Nb)+k1/2);
M     = vk2/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vk2^2*cdomsl*Sref/2;
tk2   = tg(1)+dtg/2;
Wk2   = Wg0;
k2    = dtg*(T-drag)*g/Wk2;
vk3   = (v(Nb)+k2/2);
M     = vk3/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vk3^2*cdomsl*Sref/2;
k3    = dtg*(T-drag)*g/Wk2;
vk4   = v(Nb)+k3;
M     = vk4/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vk3^2*cdomsl*Sref/2;
k4    = dtg*(T-drag)*g/Wg0;
ag(1) = k1/dtg;
vg(1) = v(Nb)+(k1+2*k2+2*k3+k4)/6;
vavg  = (v(Nb)+vg(1))/2;
distg(1) = dist(Nb)+vavg*dtg;

M     = vg(1)/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vg(1)^2*cdomsl*Sref/2;
k1    = dtg*(T-drag)*g/Wg0;
vk2   = (vg(1)+k1/2);
M     = vk2/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vk2^2*cdomsl*Sref/2;
tk2   = tg(2)+dt/2;
Wk2   = Wg0;
k2    = dtg*(T-drag)*g/Wk2;
vk3   = (vg(1)+k2/2);
```

```
M     = vk3/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vk3^2*cdomsl*Sref/2;
k3    = dtg*(T-drag)*g/Wk2;
vk4   = vg(1)+k3;
M     = vk4/sofs;
cdomsl= cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag  = rho*vk3^2*cdomsl*Sref/2;
k4    = dtg*(T-drag)*g/Wg0;
ag(2) = k1/dtg;
vg(2) = vg(1)+(k1+2*k2+2*k3+k4)/6;
distg(2)= distg(1)+(3*vg(1)-v(Nb))*dtg/2;

M       = vg(2)/sofs;
cdomsl  = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
drag    = rho*vg(2)^2*cdomsl*Sref/2;
ag(3)   = (T-drag)*g/Wg0;
vg(3)   = (3*ag(2)-ag(1))*dtg/2+vg(2);
distg(3)= distg(2)+(23*vg(2)-16*vg(1)+5*v(Nb))*dtg/12;

for k = 4:length(t)
      M       = vg(k-1)/sofs;
      cdomsl  = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,1);
      drag    = rho*vg(k-1)^2*cdomsl*Sref/2;
      ag(k)   = (T-drag)*g/Wg0;
      vg(k)   = vg(k-1)+(23*ag(k-1)-16*ag(k-2)+5*ag(k-3))*dtg/12;
      distg(k)= distg(k-1)+(23*vg(k-1)-16*vg(k-2)+5*vg(k-3))*dtg/12;
end

toc

figure(1)
clf
plot(t,v,tg,vg)
title('Velocity Versus Time - PROPEX3')
xlabel('Time (sec)')
ylabel('Velocity (ft/sec)')

figure(2)
clf
plot([t,tg],[dist/6076.11549,distg/6076.11549])
title('Range Versus Time - PROPEX3')
xlabel('Time (sec)')
ylabel('Range (NM)')

figure(3)
clf
plot([t,tg],[a,ag]/g)
title('Acceleration Versus Time - PROPEX3')
xlabel('Time (sec)')
ylabel('Acceleration (g)')
```

PROPEX4

```
function propex4(N)
%  PROPEX4 Demonstrates a boost-glide-boost-glide trajectory.  The user
%     can modify the missile and flight parameters inside propex4 for
%     use in a customized m-file.
%
%           propex4(N)
%
%           N     = Number of iterations, the default = 100;

tic

if nargin < 1
      N = 100;
end

% Missile aerodynamic data
d      = 0.6;              % Missile diameter
l      = 12;               % Missile length
ln     = 2*d;              % Nose length
bw     = 2.5817;           % Wing Span
Sw     = 2.3798;           % Wing area
thickw= 0.04;              % Wing thickness
bt     = 1.2950;           % Tail span
St     = 0.5988;           % Tail area
thickt= 0.04;              % Tail thickness
nf     = 0;                % Friction factor

% Initial altitude and velocity
Alt0   = 10000;     % Initial altitude
Vi     = 400;       % Initial velocity
Maxalt   = 50000;   % Missile ceiling.  Missile will climb at the entered
                    % flight path angles until the ceiling is reached.
                    % The flight path angle then becomes FPA = 0

% Flight parameters
%     In the following format:
%     [Thrust, burn time, weight initial, propellant weight, flight
%           path angle (DEG)]

FltPara= [10000,1.825,339,70,30;
              0,   10,269, 0,30;
          10000,1.825,269,70,30;
              0,  100,199, 0,30];

%*********************USER NEED NOT EDIT BELOW THIS LINE
g      = 32.174;
Sref   = pi*d^2/4;

[m,n] = size(FltPara);
acel   = zeros(m,N);
vel    = zeros(m,N);
dist   = zeros(m,N);
time   = zeros(m,N);
alt    = zeros(m,N);
TP     = zeros(m,N);

for j = 1:m
```

```
Para   = FltPara(j,:);
Thrust       = Para(1);
tb     = Para(2);
WL     = Para(3);
WP     = Para(4);
FPA    = Para(5);

if j == 1
       v0       = Vi;
       dist0    = 0;
       t0       = 0;
       alt0     = Alt0;
else
       v0     = vel(j-1,N);
       dist0 = dist(j-1,N);
       t0     = time(j-1,N);
       alt0   = alt(j-1,N);
end

Ceiling = alt0;



if Ceiling >= Maxalt
       FPA = 0;
end

FPA = FPA*pi/180;

if Thrust == 0
       pwr = 1;
else
       pwr = 0;
end

time(j,:) = linspace(t0,t0+tb,N);
dt       = time(j,2)-time(j,1);

data       = stan62(alt0);
sofs       = data(2);
rho        = data(3);
M          = v0/sofs;
cdomsl     = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,pwr);
drag       = 0.5*rho*v0^2*cdomsl*Sref;
W          = WL-WP*(time(j,1)-t0)/tb;
acel(j,1)= (Thrust-drag)*g/W-g*sin(FPA);
vel(j,1)  = acel(j,1)*dt+v0;
dist(j,1)= dist0+(v0+vel(j,1))*dt*cos(FPA)/2;
alt(j,1)  = alt0 + vel(j,1)*sin(FPA);
if alt(j,1) > Maxalt
       FPA      = 0;
       Ceiling = Maxalt;
       alt(j,1)= Maxalt;
end
TP(j,1)    = Thrust;

data       = stan62(alt(j,1));
sofs       = data(2);
rho        = data(3);
M          = vel(j,1)/sofs;
cdomsl     = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,pwr);
```

```
      drag      = 0.5*rho*vel(j,1)^2*cdomsl*Sref;
      W         = WL-WP*(time(j,2)-t0)/tb;
      acel(j,2)= (Thrust-drag)*g/W-g*sin(FPA);
      vel(j,2) = (acel(j,1)+acel(j,2))*dt/2+v0;
      dist(j,2)= dist(j,1)+(3*vel(j,1)-v0)*dt*cos(FPA)/2;
      alt(j,2) = alt(j,1) + vel(j,2)*sin(FPA);
      if alt(j,2) > Maxalt
            FPA      = 0;
            Ceiling  = Maxalt;
            alt(j,2) = Maxalt;
      end
      TP(j,2)   = Thrust;

      data      = stan62(alt(j,2));
      sofs      = data(2);
      rho       = data(3);
      M         = vel(j,2)/sofs;
      cdomsl    = cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,pwr);
      drag      = 0.5*rho*vel(j,2)^2*cdomsl*Sref;
      W         = WL-WP*(time(j,3)-t0)/tb;
      acel(j,3)= (Thrust-drag)*g/W-g*sin(FPA);
      vel(j,3) = vel(j,2)+(3*acel(j,2)-acel(j,1))*dt/2;
      dist(j,3)=dist(j,2)+(23*vel(j,2)-16*vel(j,1)+5*v0)*cos(FPA)*dt/12;
      alt(j,3) = alt(j,2) + vel(j,3)*sin(FPA);
      if alt(j,3) > Maxalt
            FPA      = 0;
            Ceiling = Maxalt;
            alt(j,3)= Maxalt;
      end
      TP(j,3)    = Thrust;

      for k = 4:N

            data    = stan62(alt(j,k-1));
            sofs    = data(2);
            rho     = data(3);
            M       = vel(j,k-1)/sofs;
            cdomsl =
                  cdowbt(d,l,ln,bw,Sw,thickw,bt,St,thickt,M,alt,nf,pwr);
            drag    = 0.5*rho*vel(j,k-1)^2*cdomsl*Sref;
            W       = WL-WP*(time(j,k)-t0)/tb;
            acel(j,k)= (Thrust-drag)*g/W-g*sin(FPA);
            vel(j,k)=vel(j,k-1)+(23*acel(j,k-1)-16*acel(j,k
                              -2)+5*acel(j,k-3))*dt/12;
            dist(j,k)=dist(j,k-1)+(23*vel(j,k-1)-16*vel(j,k
                              -2)+5*vel(j,k-3))*cos(FPA)*dt/12;
            alt(j,k) = alt(j,k-1) + vel(j,k)*sin(FPA);
            if alt(j,k) > Maxalt
                  FPA = 0;
                  Ceiling = Maxalt;
                  alt(j,k) = Maxalt;
            end
            TP(j,k) = Thrust;

      end

      toc

end
```

```
figure(1)
clf
plot(time',vel')
title('Velocity profile')
xlabel('Time (sec)')
ylabel('Velocity (ft/sec)')

figure(2)
clf
plot(time',dist'/6076.11549)
title('Range Plot')
xlabel('Time (sec)')
ylabel('Range (NM)')

figure(3)
clf
plot(time',acel'/g)
title('Acceleration')
xlabel('Time (sec)')
ylabel('Acceleration (g)')

t = 0;
tp = 1.2*max(max(alt));
figure(4)
clf
plot(t,tp/1000,time',alt'/1000)
title('Altitude Profile')
xlabel('Time (sec)')
ylabel('Altitude (10^3 ft)')


t = 0;
tp = 1.2*max(max(max(TP)),max(max(vel)));
figure(5)
clf
plot(t,tp,time',TP',time',vel')
title('Thrust-Velocity Profile')
xlabel('Time (sec)')
ylabel('Thrust (lb) - Velocity (ft/sec)')
```

REGRESS1

```
function [a,b,r] = regress1(x,y,alpha)
%  REGRESS1 Calculates the coefficients for multiplicative regression.
%
%           [a,b,r]=regress1(x,y,alpha)
%
%           y = a*x^b
%

[m,n] = size(x);

if m < n
      x = x';
end

[m,n] = size(y);

if m < n
      y = y';
end

x1 = log(x);
y1 = log(y);

X = ones(size(x));

X = [X x1];

[b1,bint,r,rint,stats]=regress(y1,X,alpha);

a = exp(b1(1));
b = b1(2);

r = stats(1);
```

REYNOLDS

```
function rn = reynolds(chardim,Mach,altitude,iunits)
%  REYNOLDS Calculates the reynolds number for a standard atmosphere.
%
%          rn = reynolds(chardim,Mach,altitude,iunits)
%
%           chardim  = Characteristic dimension
%           Mach     = Mach number
%           altitude = altitude
%
%     If iunits is omitted, English units are the default.  If
%     metric units are required use iunits = 1.

if nargin == 4;
        g = 9.80665;
      iunits = 1;
else,
        g = 32.1741;
      iunits = 2;
end

x = stan62(altitude,iunits);        % Uses stan62 for atmospheric data

rn = x(3)*Mach*x(2)*chardim*g/x(6);
```

RGRSDEMO

```
function rgrsdemo()
%  RGRSDEMO demonstrates investigating the relationship between air-to-
%     air missile volume to weight using simple linear regression.
%
%     See also AAMDWW, RGRSSLIN and RGRSSPOW.

disp(' ')
disp('RGRSDEMO investigates the relationship between air-to-air missile
volume (VOL) to')
disp('weight (WT) using simple linear regression models.  Two
relationships will be examined:')
disp(' ')
disp('                          WT = b1*VOL + b0')
disp('                                 and              ')
disp('                            WT = a*VOL^b')
disp(' ')
disp('The m-files RGRSSLIN and RGRSSPOW will be applied to the data from
the m-file AAMDWW.')

data = aamdww;

WT = data(:,2);
VOL = data(:,3);

[b1,b0,r2] = rgrsslin(VOL,WT);

[a,b,r] = rgrsspow(VOL,WT);


lin = [b1 b0 r2];
pow = [a b r];

disp(' ')
disp('       Coefficients')
disp('       b1         b0         r^2')
disp(lin)
disp('       a          b')
disp(pow)
disp(' ')
disp('Thus, the power law model correlates these particular data')
disp('better than does the linear model.')
```

RGRSSLIN

```
function [b1,b0,r2]=rgrsslin(x,y)
%  RGRSSLIN Calculates the coefficients for linear regression and the
%     coefficient of determination, r^2.
%
%                  [b1,b0,r2] = rgrsslin(x,y)
%
%                  y = b1*x + b0 + e
%
%                  e = normally distributed error


n = length(x);

b1 = (n*sum(x.*y)-sum(x)*sum(y))/(n*sum(x.^2)-sum(x)^2);

b0 = sum(y)/n-b1*sum(x)/n;

%r = n*sum(x.*y)-sum(x)*sum(y);

%r = r/((n*sum(x.^2)-sum(x)^2)*(n*sum(y.^2)-sum(y)^2))^0.5;

r2 = 1-(sum(y.^2)-b0*sum(y)-b1*sum(x.*y))/(sum(y.^2)-(sum(y)^2)/n);
```

RGRSSPOW

```
function [a,b,r2]=rgrsspow(x,y)
%  RGRSSPOW Calculates the coefficients for a power relationship by
%     transforming the power relationship into a linear relationship.
%     The coefficient of determination, r2, is also returned.
%
%                  [a,b,r2] = rgrsspow(x,y)
%
%                  y = a*x^b


x1 = log(x);
y1 = log(y);

[b1,b0,r2] = rgrsslin(x1,y1);

a = exp(b0);
b = b1;
```

SMANDLF

```
%********************SMANDLF********************
%
%  SMANDLF is a sample m-file which generates plots of the static
%     margin and load factor capability

%  Define the missile parameters
d = 0.6;
l = 12;
ln = 2*d;
lcgi = 6.5553;
lcgf = 4.7014;
bw = 2.5817;
Sw = 2.3798;
lw = 5.5767;
bt = 1.295;
St = 0.598;
lcgi = 6.5553;
lcgf = 4.7014;
Wi = 339;
Wf = 200;
alt = 10000;
alpha = 0;
Sref = pi*d^2/4;
g = 32.174;

Mach =
[0.1,.2,.4,.6,.8,.95,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9,2,2.3,2.6,2.75,3];
cmamsl = zeros(size(Mach));
clamsl = zeros(size(Mach));
cmdmsl = zeros(size(Mach));
cldmsl = zeros(size(Mach));
smi = zeros(size(Mach));
lfi = zeros(size(Mach));

for k = 1:length(Mach)
      cmamsl(k) = cmawbt(d,l,ln,lcgi,bw,Sw,lw,bt,St,Mach(k),alpha);
      clamsl(k) = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach(k),alpha);
      cmdmsl(k) = cmdwbt(d,l,lcgi,bt,St,Mach(k));
      cldmsl(k) = cldwbt(d,bt,St,Mach(k));
      qbar = dynmprss(Mach(k),alt);
      lfi(k) = (cldmsl(k)-clamsl(k)*cmdmsl(k)/cmamsl(k))*qbar*Sref/Wi/g;
end

smi = cmamsl./clamsl/d;

smf = zeros(size(Mach));
lff = zeros(size(Mach));

for k = 1:length(Mach)
      cmamsl(k) = cmawbt(d,l,ln,lcgf,bw,Sw,lw,bt,St,Mach(k),alpha);
      clamsl(k) = clawbt(d,l,ln,bw,Sw,lw,bt,St,Mach(k),alpha);
      cmdmsl(k) = cmdwbt(d,l,lcgi,bt,St,Mach(k));
      cldmsl(k) = cldwbt(d,bt,St,Mach(k));
      qbar = dynmprss(Mach(k),alt);
      lff(k) = (cldmsl(k)-clamsl(k)*cmdmsl(k)/cmamsl(k))*qbar*Sref/Wf/g;
end

smf = cmamsl./clamsl/d;
```

```
figure(1)
clf
plot(Mach,smi,Mach,smf)
title('Static Margin versus Mach Number')
xlabel('Mach Number')
ylabel('Static Margin in Fractions of Diameter')
text(0.5,-1.5,'Begining of Flight')
text(1.5,-5,'End of Flight')

figure(2)
clf
plot(Mach,lfi,Mach,lff)
title('Load Factor Capability versus Mach Number')
xlabel('Mach Number')
ylabel('Load Factor Capability per Unit Deflection (RAD)')
text(1.5,-25,'Beginning of Flight')
text(1.5,-3,'End of Flight')
```

SMLKBW

```
function kbw = smlkbw(r,s)
%  SMLKBW Calculates kb(w)the lift of the body in the presence of the
%     wing for variable incidence.
%
%                  kbw = smlkbw(r,s)
%
%                  r = radius of the body
%                  s = span = radius + exposed semi-span

%     David A. Ekker from NACA 1307

kbw = bigkwb(r,s)-smlkwb(r,s);
```

SMLKWB

```
function kwb = smlkwb(r,s)
%  SMLKWB Calculates kw(b) the lift of the wing in the presence of the
%     body for variable incidence.
%
%                  kwb = smlkwb(r,s)
%
%                  r = radius of the body
%                  s = span = radius + exposed semi-span

%     David A. Ekker

t = s/r;

if t <= 1
      kwb = 1;
else
      kwb = pi^2*(t+1)^2/4/t^2+pi*(t^2+1)^2*asin((t^2-
1)/(t^2+1))/t^2/(t-1)^2;
      kwb = kwb-2*pi*(t+1)/t/(t-1)+(t^2+1)^2*(asin((t^2-
1)/(t^2+1)))^2/t^2/(t-1)^2;
      kwb = kwb-4*(t+1)*asin((t^2-1)/(t^2+1))/t/(t-
1)+8*log((t^2+1)/2/t)/(t-1)^2;
      kwb = kwb/pi^2;
end
```

SSCFDCCC

```
function cdc = sscfdccc(alpha,M)
%  SSCFDCCC Calculates the steady-state cross-flow drag coefficient for
%     circular cylinders (two dimensional)
%
%                  cdc = sscfdccc(alpha,M)
%
%                  alpha = angle of attack (rad)
%                  M     = Mach number

%     Written by David A. Ekker
%     Derived from Figure 4.2.1.2-35b of USAF Stability and Control
%     Datcom Global Engineering Documents, April 1978

pa = [-2.5513    4.9961   -1.5000    0.1394    1.1967];
pb = [-2.1943e+2,9.0057e+2,-1.4619e+3,1.1695e+3,-4.5889e+2,7.1919e+1];
pc = [154.7931 -553.2488  737.6712 -434.1301   96.7047];
pd = [-3.4916,7.9077,-6.1199,2.3609,-7.8519e-2,1.2400];

[m,n] = size(alpha);

if n > m
      alp = alpha';
else
      alp = alpha;
end

[m,n] = size(M);

if m > n
      M1 = M';
else
      M1 = M;
end

Mc = sin(alp)*M1;

[m,n]=size(Mc);
cdc = zeros(m,n);
n = m*n;

for l = 1:n
      k = Mc(l);

      if k > 1
            k = 1/k;

            if k > 0.8
                  cdc(l) = polyval(pc,k);
            else
                  cdc(l) = polyval(pd,k);
            end
      else
            if k > 0.6
                  cdc(l) = polyval(pb,k);
            else
                  cdc(l) = polyval(pa,k);
            end
      end
```

```
end
```

STAN62

```
function X = stan62(height,iunits)
%  STAN62 Calculates the atmospheric properties based on a U.S. standard
%     model atmosphere, 1962 prepared by AMSMI-RST, 21 July 1962.
%
%           data = stan62(altitude,iunits)
%
%  for the input altitude.  English units are default.  If metric
%  units are required use stan62(altitude,1).
%
%           data = [altitude, speed of sound, density, pressure,
%                 temperature, coefficient of viscosity]
%
%                                  English       Metric
%
%           altitude          feet           meters
%           speed of sound    ft/sec         m/sec
%           density             slug/ft^3      kg/m^3
%           pressure      lb/ft^2        kg/m^2
%           temperature       R              K
%           viscosity     lbm/ft-sec     kg/sec-m
%
%  This function is valid below 700,000 meters.  Extrapolation is
%  provided above 700,000 meters.
%
%     See also ATMSCOFV, ATMSDENS, ATMSPRSS, ATMSSOFS and ATMSTEMP.

height = height(1,1);

if nargin < 2; iunits = 2; end;          %  Default to English units

if height < 0; height = 0; end;          %  Default altitude to a 0
minimum

h = height;

if iunits ~= 1,                          %  All calculations are in
metric
        h = h*0.3048;                    %  convert to meters
end

z = h*0.001;

%  *************************     0 - 90,000 meters

if z <= 90.0,
        p = [-9.5013649e-35 6.0621354e-28 -3.8667054e-21 2.4656553e-14 -
1.5731262e-7 1 0];
        h = polyval(p,h)*0.001;
        hh = [11.0 20.0 32.0 47.0 52.0 61.0 79.0 88.743];

        ind = 1;
        while  hh(ind) < h
                ind = ind + 1;
        end

        tbh = [288.15 216.65 216.65 228.65 270.65 270.65 252.65 180.65];
        hbh = [0 11 20 32 47 52 61 79];
        rlh = [-6.5 0 1 2.8 0 -2 -4 0];
```

```
        pbh = [10331.9076 2307.7398 558.2615 88.50965 11.30876 6.016163
               1.85682 0.105812];

        if (ind ~= 2 & ind ~= 5 & ind ~= 8),
                temp = tbh(ind)+(h-hbh(ind))*rlh(ind);
                press = pbh(ind)*(tbh(ind)/temp)^(34.16479/rlh(ind));
        else
                temp = tbh(ind);
                press = pbh(ind)*exp(-34.16479*(h-hbh(ind))/tbh(ind));
        end

        rho = press/(29.26945*temp);
        vsound = 20.046333*temp^0.5;

%  ************************** 90,000 - 700,000 meters

else
        z1 = height;

        if z1 <= 170000,
                xm = (((((0.14186509e-27*z1-0.111458341e-
21)*z1+0.359201416e-16)*z1-0.60665213e-11)*z1+0.565215183e-6)*z1-
0.275300356e-1)*z1+5.76957191e+2;
        else
                xm = (((((1.180554e-33*z1-3.429162e-27)*z1+3.8159669e-
21)*z1-2.0402054e-15)*z1+5.6422141e-10)*z1-1.0700489e-
4)*z1+3.5629995e+1;
        end

        if z > 700,
                ind2 = 13;
                disp('Altitude greater than 700000 meters,
extrapolating')
        else
                zz = [100 110 120 150 160 170 190 230 300 400 500 600
700];

                ind2 = 1;
                while zz(ind2) < z
                        ind2 = ind2 + 1;
                end
        end

        tb = [180.65 210.65 260.65 360.65 960.65 1110.65 1210.65 1350.65
1550.65 1830.65 2160.65 2420.65 2590.65];
        zb = [90 100 110 120 150 160 170 190 230 300 400 500 600];
        rl = [3 5 10 20 15 10 7 5 4 3.3 2.6 1.7 1.1];
        pb = [0.016762 0.0030667 0.7499e-3 0.2571e-3 0.516e-4 0.3767e-4
0.28476e-4 0.17184e-4 0.709767e-5 0.1921e-5 0.41097e-6 0.1117e-6
0.3518e-7];
        xxx = [33.166119 33.064396 32.96512 32.784318 32.530427
32.455229 32.309443 32.016142 31.492584 30.704189 29.808321 28.937691
28.123252];

        tm = tb(ind2) + (z-zb(ind2))*rl(ind2);
        temp = tm*xm/28.964;
        press = pb(ind2)*(tb(ind2)/tm)^(xxx(ind2)/rl(ind2));
        rho = press/(29.26945*tm);
        vsound = (401.90467*tm)^0.5;

end
```

```
%  Convert back to English units if necessary

u = 1.458e-6*temp^1.5/(temp + 110.4);

if iunits == 2,
        vsound = vsound*3.2808333;
        rho = rho*0.194032e-2;
        press = press*0.20482;
        temp = temp*1.8;
        u = 7.3025e-7*temp^1.5/(temp + 198.72);
end

X = [height vsound rho press temp u];
```

STPRSPNS

```
function [r,tp,Mp] = stprspns(t,wn,zeta)
%  STPRSPNS returns the normalized magnitude of the step response for a
%     second order system of the form s^2+2*wn*zeta*s+wn^2=0.
%
%           [r,tp,Mp] = stprspns(t,wn,zeta)
%
%           r     = response at time(sec)
%           tp    = time to peak.  If zeta >= 1 this value is NaN
%           Mp    = maximum overshoot
%           t     = time vector
%           wn    = natural frequency
%           zeta  = damping factor

r = zeros(length(t));

if zeta < 1
      phi = atan(zeta/sqrt(1-zeta^2));
      wd = wn*sqrt(1-zeta^2);
      r = 1 - exp(-zeta*wn*t).*cos(wd*t+phi)./sqrt(1-zeta^2);
      tp = pi/wd;
      Mp = exp(-pi*zeta/sqrt(1-zeta^2));
elseif zeta == 1
      r = 1 - exp(-zeta*wn*t);
      Mp = 1;
      tp = NaN;
else
      r = 1 - exp(-zeta*wn*t);
      z2 = sqrt(zeta^2-1)*wn;
      r = r+zeta*exp(-zeta*wn*t).*(exp(-z2*t)-exp(z2*t))./2/sqrt(zeta^2
          -1);
      Mp = 1;
      tp = NaN;
end
```

SURFOGV1

```
function surfW = surfogv1(diameter,length)
%  SURFOGV1 Calculates the wetted surface area of an ogive less the
%     area of the base.
%
%           surfW = surfogv1(diameter,length)
%
%           surfW = wetted surface area less the base
%           length      = length of ogive
%           diameter= base diameter

%     David A. Ekker

ldr = length./diameter;

surf1 = zeros(size(ldr));

ldr2 = ldr.^2;

surf1 = asin(ldr./(ldr2 + 0.25));

surfW = pi*diameter^2*((ldr2 + 0.25).^2.*surf1 - ldr.*(ldr2 - 0.25));
```

TLSZCN1

```
function [bt,St,lt] = tlszcn1(d,l,lcg,M,bw,Sw,lw,SWPtail,amax,dmax)
%  TLSZCN1 Returns the tail dimensions and location for a tail-
%     controlled missile to provide stability at amax with a dmax tail
%     deflection.
%
%           [bt,St,lt] = tlszcn1(d,l,lcg,M,bw,Sw,lw,SWPtail,amax,dmax)
%
%           bt     = tail span
%           St     = tail size
%           lt     = tail location measured from the tip of the nose
%           d      = missile diameter
%           l      = missile length
%           lcg    = location of center of gravity measured from the nose
%           M      = Mach number
%           bw     = wing span
%           Sw     = wing size
%           lw     = wing location measured from the tip of the nose
%           SWPtail      = tail leading edge sweep angle (DEG)
%           amax   = missile maximum angle of attack (DEG).  15 degrees
%                    is assumed if no value is entered.
%           dmax   = tail deflection maximum angle (DEG).  15 degrees is
%                    assumed if no value is entered.
%

if nargin < 9
      amax = 15*pi/180;
      dmax = 15*pi/180;
elseif nargin < 10
      amax = amax*pi/180;
      dmax = 15*pi/180;
else
      amax = amax*pi/180;
      dmax = dmax*pi/180;
end

tol = 0.0001;
SWPtail = SWPtail*pi/180;
Sref = pi*(d/2)^2;

%     Calculate the wing parameters

Aw = bw^2/Sw;
crw = 2*Sw/bw;
mw = bw/2/crw;
sw = 0.5*(d+bw);
betaw = sqrt(abs(M^2-1));
Kbw = bigkbwa(M,Aw,d/2,sw);
Kwb = bigkwb(d/2,sw);
CLaW = clawsup(M,Aw);
xcrwb = xcrwba(d/2,sw);
SWPwing = atan(2*crw/bw);

if betaw*Aw >= 0
      xcrbw = xcrbwabh(M,d/2,crw);
else
      xcrbw = xcrbwabl(M,Aw,d/2,sw);
end
```

```
xbw = lw+xcrbw*crw-lcg;
xwb = lw+xcrwb*crw-lcg;

CLW = (Kbw*xbw+Kwb*xwb)*CLaW*Sw;

%     Calculate the center of pressure of the nose

ln = 2*d;
cpn = cpogvemp(d,ln,M);
XN = lcg-cpn;
Ap = (1-ln)*d+areapogv(d,ln);
cdc = sscfdccc(amax,M);
CLaN = 2*k2mk1(l/d)+cdc*Ap*amax/Sref;
CLN = CLaN*XN*Sref;

Mt = 0.95*M;
nt = (Mt/M)^2;
St0 = tol;
betat = sqrt(Mt^2-1);

NW = (CLN-CLW)*amax/nt;

if NW >= 0

      dSt = 1000;
      inc = 1.0;
      k = 0;

      while dSt > tol

            dSt = 1000;
            inc = 0.5*inc;

            while dSt > 0

                  St1 = St0+k*inc;
                  St = St1;
                  bt = (4*St/tan(SWPtail));
                  At = bt^2/St;
                  crt = 2*St/bt;
                  mt = bt/2/crt;
                  st = 0.5*(d+bt);
                  Kbt = bigkbwna(Mt,At,d/2,st);
                  Ktb = bigkwb(d/2,st);
                  CLaT = clawsup(Mt,At);
                  xcrtba = xcrwba(d/2,st);
                  xcrtbd = xcrwbd(d/2,st);
                  kbt = smlkbw(d/2,st);
                  ktb = smlkwb(d/2,st);


                  if betat*At >= 0
                        xcrbt = xcrbwnab(Mt,d/2,crt);
                  else
                        xcrbt = xcrbwabl(Mt,At,d/2,st);
                  end

                  lt = 1-crt;
                  xbt = lt+xcrbt*crt-lcg;
                  xtba = lt+xcrtba*crt-lcg;
                  xtbd = lt+xcrtbd*crt-lcg;
```

```
                    x0 = (lt-(lw+crw))/crw;

                    deda = Kwb*dedasup(M,SWPwing,x0);

                    dSt = NW-CLaT*((Kbt*xbt+Ktb*xtba)*(1-
deda)*amax+(kbt*xbt+ktb*xtbd)*dmax)*St;

                    if abs(dSt) < tol
                          dSt = abs(dSt);
                          break;
                    end

                    k = k+1;

                    if k > 20
                          break;
                    end

              end

              St0 = St0+(k-2)*inc;

              if dSt < 0
                    dSt = abs(dSt);
              end

              if dSt < tol
                    break;
              end

              k = 1;

        end

else

        dmax = -dmax;
        dSt = 1000;
        inc = 1.0;
        k = 0;

        while dSt > tol

              dSt = -1000;
              inc = 0.5*inc;

              while dSt < 0

                    St1 = St0+k*inc;
                    St = St1;
                    bt = (4*St/tan(SWPtail));
                    At = bt^2/St;
                    crt = 2*St/bt;
                    mt = bt/2/crt;
                    st = 0.5*(d+bt);
                    Kbt = bigkbwna(Mt,At,d/2,st);
                    Ktb = bigkwb(d/2,st);
                    CLaT = clawsup(Mt,At);
                    xcrtba = xcrwba(d/2,st);
                    xcrtbd = xcrwbd(d/2,st);
```

```
                    kbt = smlkbw(d/2,st);
                    ktb = smlkwb(d/2,st);

                    if betat*At >= 0
                          xcrbt = xcrbwnab(Mt,d/2,crt);
                    else
                          xcrbt = xcrbwabl(Mt,At,d/2,st);
                    end

                    lt = l-crt;
                    xbt = lt+xcrbt*crt-lcg;
                    xtba = lt+xcrtba*crt-lcg;
                    xtbd = lt+xcrtbd*crt-lcg;

                    x0 = (lt-(lw+crw))/crw;

                    deda = Kwb*dedasup(M,SWPwing,x0);

                    dSt = NW-CLaT*((Kbt*xbt+Ktb*xtba)*(1-
deda)*amax+(kbt*xbt+ktb*xtbd)*dmax)*St;

                    if abs(dSt) < tol
                          dSt = abs(dSt);
                          break;
                    end

                    k = k+1;

                    if k > 20
                          break;
                    end

              end

              St0 = St0+(k-2)*inc;

              if dSt < 0
                    dSt = abs(dSt);
              end

              if dSt < tol
                    break;
              end

              k = 1;

        end

end
```

TLSZEX1

```
%******************** TLSZEX1 ********************
%           by David A. Ekker
%
%  TLSZEX1 is a script work file used in MSLDSN for defining
%  the optimum tail size for a tail-controlled MRAAM.

%  Calculate the initial center of gravity, lcgi from AMRAAM data.
lcgi = xcgtc1(5.88,120,0.92,44,4.89,154,21.68);

%  Input missile data
d = 0.6;     %  Missile diameter
l = 12;          %  Missile length
W = 339;     %  Missile weight
nm = 31;     %  Load Factor
alt = 10000;      %  Altitude
SWPwing = 55;     %  Sweep angle

%  Calculate the required wing sizes for the Mach 1.7
M = 1.7;
[bw17,Sw17,amax17]=wngsztc1(d,l,W,M,nm,alt,SWPwing);

%  Calculate the center of gravity at the end and one half missile burn
life
lcgf = xcgtc1(5.88,120,0.92,44,4.89,0.1*154,21.68);

%  Calculate the wing position at beginning of missile burn
lwi = wngpos1(d,l,W,1.7,lcgi,alt,bw17,Sw17);

%  Calculate the wing position at the end of missile burn
Wf = W - 0.9*154;
lwf = wngpos1(d,l,Wf,1.7,lcgf,alt,bw17,Sw17);

%  Define the range of wing positions to check.  Since the m-file
WNGPOS1 assumes
%  an initial tail size of 0.4*Sw, the allowable range of wing position
locations
%  is slightly larger than the lwi calculation.
lw = linspace(lwf,lwi+1,15);

%  Set up the variables required.
bts = size(lw);
Sts = size(lw);
lts = size(lw);
btci17 = size(lw);
Stci17 = size(lw);
ltci17 = size(lw);
btcf17 = size(lw);
Stcf17 = size(lw);
ltcf17 = size(lw);
btci12 = size(lw);
Stci12 = size(lw);
ltci12 = size(lw);
btcf12 = size(lw);
Stcf12 = size(lw);
ltcf12 = size(lw);

SWPtail = 55;
tic
```

```
for k = 1:length(lw)

      itrtn = length(lw)+1 - k

      %  Check the tail size for stability
      [bts(k),Sts(k),lts(k)]=tlszstb1(d,l,lcgi,bw17,Sw17,lw(k),SWPtail);

      %  Check the tail size for controllability at the beginning and
end of burn
      [btci17(k),Stci17(k),ltci17(k)]=tlszcn1(d,l,lcgi,1.7,bw17,Sw17,lw(
k),SWPtail);
      [btcf17(k),Stcf17(k),ltcf17(k)]=tlszcn1(d,l,lcgf,1.7,bw17,Sw17,lw(
k),SWPtail);

      %  Check the tail size for controllability at the beginning and
end of burn
      [btci12(k),Stci12(k),ltci12(k)]=tlszcn1(d,l,lcgi,1.2,bw17,Sw17,lw(
k),SWPtail);
      [btcf12(k),Stcf12(k),ltcf12(k)]=tlszcn1(d,l,lcgf,1.2,bw17,Sw17,lw(
k),SWPtail);

      toc
end

clf
plot(lw,Sts,lw,Stci17,lw,Stcf17,lw,Stci12,lw,Stcf12,5.73,.58,'o')
title('Tail Size Versus Wing Position')
xlabel('Wing Location Measured From the Nose')
ylabel('Tail Size')
text(5.5,2.85,'BOF: Beginning of Flight')
text(5.5,2.7,'EOF: End of Flight')
text(5.5,2.55,'OPT: Optimum Point')
text(4,2.5,'Stability')
text(3.7,1.7,'Mach 1.7 BOF')
text(5.9,2.1,'Mach 1.7 EOF')
text(3.7,1,'Mach 1.2 BOF')
text(4.4,.35,'Mach 1.2 EOF')
text(5.6,0.8,'OPT')

%  Find and return the optimum tail size and wing location

Stmtrx = max([Sts;Stci17;Stcf17;Stci12;Stcf12]);
[Stopt,index] = min(Stmtrx);

btopt = sqrt(4*Stopt/tan(SWPtail*pi/180))
Stopt
lwopt = lw(index)

pause

bt = sqrt(4*Stopt/tan(SWPtail*pi/180));
crt = 2*Stopt/bt;
crw = 2*Sw17/bw17;

figure(2)
msldsply(d,l,2*d,lwopt,crw,bw17,0,SWPwing,l-crt,crt,bt,0,SWPtail)
title('TLSZEX1 Missile')
figure(1)
```

TLSZEX2

```
%******************** TLSZEX2 ********************
%           by David A. Ekker
%
%  TLSZEX2 is a script working m-file used in MSLDSN for defining
%  the optimum tail size for a tail-controlled MRAAM with an
%  initial tail size other than 0.4*Sw.

%  Input missile data
d = 0.6;     %  Missile diameter
l = 12;          %  Missile length
W = 339;     %  Missile weight
nm = 31;     %  Load Factor
alt = 10000;      %  Altitude
SWPwing = 55;     %  Sweep angle
lw0 = 5.7083;     %  Initial wing location
bt0 = 1.2988;     %  Initial tail span
St0 = 0.6023;     %  Initial tail size

%  Calculate the wing size with the default tail size
[bworig,Sworig,amaxorig]=wngsztc1(d,l,W,M,nm,alt,SWPwing);

%  Calculate the required wing sizes for the Mach 1.7
M = 1.7;
[bw17,Sw17,amax17]=wngsztc1(d,l,W,M,nm,alt,SWPwing,0,lw0,bt0,St0);

%  Calculate the initial center of gravity, lcgi from AMRAAM data.
%  The location of the wing and the tail are known.  The center
%  of gravity of a triangular planform is 2/3 of the root chord
%  measured from the apex.  The weight of the wing and tail come
%  from the wing-to-tail size ratio.

crw = 2*Sw17/bw17;
crt = 2*St0/bt0;
tlcg = l-crt/3;
wngcg = lw0+2*crw/3;
wfwt = 21.68;
wngwt = wfwt/(1+St0/Sw17);
tlwt = St0*wngwt/Sw17;

lcgi = xcgtc2(5.88,120,0.92,44,4.89,154,wngwt,wngcg,tlwt,tlcg);

%  Calculate the center of gravity at the end and one half missile burn
life
lcgf = xcgtc2(5.88,120,0.92,44,4.89,0.1*154,wngwt,wngcg,tlwt,tlcg);

%  Define the range of wing positions to check.  Increasing the number
%  of increments (from the present 20) may help the precision of the
%  answer but will increase the calculation time.
lw = linspace(lw0-0.5,lw0+0.5,20);

%  Set up the variables required.
btci17 = size(lw);
Stci17 = size(lw);
ltci17 = size(lw);
btcf17 = size(lw);
Stcf17 = size(lw);
ltcf17 = size(lw);
```

```
SWPtail = 55;
tic
for k = 1:length(lw)

      itrtn = length(lw)+1 - k

      %  Check the tail size for controllability at the beginning and
end of burn
      [btci17(k),Stci17(k),ltci17(k)]=tlszcn1(d,1,lcgi,1.7,bw17,Sw17,lw(
k),SWPtail);
      [btcf17(k),Stcf17(k),ltcf17(k)]=tlszcn1(d,1,lcgf,1.7,bw17,Sw17,lw(
k),SWPtail);

      toc
end

Stmtrx = max([Stci17;Stcf17]);
[Stopt,index] = min(Stmtrx);

btopt = sqrt(4*Stopt/tan(SWPtail*pi/180));
Stopt;
lwopt = lw(index);

dSw = abs(Sw17-Sworig)
dSt = abs(Stopt-St0)

%  Find and return the optimum tail size and wing location

bwopt = bw17
Swopt = Sw17
lwopt
btopt
Stopt

bt = btopt;
crt = 2*Stopt/bt;
crw = 2*Sw17/bw17;

figure(1)
clf
msldsply(d,1,2*d,lwopt,crw,bw17,0,SWPwing,1-crt,crt,bt,0,SWPtail)
title('TLSZEX2 Missile')
```

TLSZSTB1

```
function [bt,St,lt]=tlszstb1(d,l,lcg,bw,Sw,lw,SWPtail)
%  TLSZSTB1 Calculates the tail area required for stability of a tail-
%     controlled missile in the subsonic regime.
%
%           [bt,St,lt]=tlszstb1(d,l,lcg,bw,Sw,lw,SWPtail)
%
%           bt     = tail span
%           St     = tail size
%           lt     = distance from nose to leading edge of tail
%           d      = missile diameter
%           l      = missile length
%           lcg    = distance from nose to center of gravity
%           bw     = wing span
%           Sw     = wing size
%           lw     = distance from nose to leading edge of wing
%           SWPtail      = tail leading edge sweep angle (DEG)

Sref = pi*(d/2)^2;

CLaN = 2*k2mk1(l/d);
cpn = cpogvsb(2*d,d);
xn = lcg-cpn;

M = 0:0.05:0.95;
[v,w]=size(M);
Stm = ones(size(M));
Stm = 0.4*Sw*Stm;
sw = 0.5*(d+bw);
Aw = bw^2/Sw;
crw = 2*Sw/bw;
kappa = 0.85;
CLaW0 = clawsub(0,Aw,atan(2/Aw),kappa);
Kbw = bigkbwsb(d/2,sw);
Kwb = bigkwb(d/2,sw);
lc4w = atan(3/Aw);

for k = 1:w

      CLaW = clawsub(M(k),Aw,atan(2/Aw),kappa);
      xbw = lw+xcrbwsub(M(k),Aw,d/2,sw)*crw-lcg;
      xwb = lw+xcrwbsub(M(k),Aw)*crw-lcg;
      NW = CLaN*xn*Sref-(Kbw*xbw+Kwb*xwb)*CLaW*Sw;

      if NW < 0
            bt = 0;
            St = 0;
            lt = 1;
            return;
      end

      Mt=0.95*M(k);
      if M(k) == 0
            nt = 1;
      else
            nt = (Mt/M(k))^2;
      end

      dSt = 10000;
```

```
        while dSt > 0.001

                bt = sqrt(4*Stm(k)/tan(SWPtail*pi/180));
                At = bt^2/Stm(k);
                crt = 2*Stm(k)/bt;
                st = 0.5*(d+bt);
                Kbt = bigkbwsb(d/2,st);
                Ktb = bigkwb(d/2,st);
                lt = l-crt;
                xbt = lt+xcrbwsub(Mt,At,d/2,st)*crt-lcg;
                xtb = lt+xcrwbsub(Mt,At)*crt-lcg;

                CLaT = clawsub(Mt,At,atan(2/At),kappa);

                lH = l-Stm(k)/bt-(lw+Sw/bw);
                dedaM = dedasub(Aw,bw,lc4w,0,lH,0)*CLaW/CLaW0;

                St1 = NW/(Kbt*xbt+Ktb*xtb)/nt/CLaT/(1-dedaM);

                dSt = abs(Stm(k)-St1);

                Stm(k) = St1;

        end

        if k ~= w
                Stm(k+1)=Stm(k);
        end
end

St = max(Stm);
bt = sqrt(4*Stm(k)/tan(SWPtail*pi/180));
crt = 2*Stm(k)/bt;
lt = l-crt;
```

VOLOGV1

```
function vol = vologv1(logv,dbase)
%  VOLOGV1 Calculates the volume of an ogive based on the length
%     of the ogive and the base diameter.  If logv is a vector,
%     vol returns a vector of volumes.  dbase must be a single
%     value

%     David A. Ekker

l = logv;
d = dbase;

R = zeros(size(l));
vol = zeros(size(l));

R = d/4 + l.^2/d;

vol=(R-d/2).*(l.*(R.^2-l.^2).^0.5 + R.^2.*asin(l./R));

vol=pi*(l.*(2*R.^2 - R*d + d^2/4) - l.^3/3 - vol);
```

WNGPOS1

```
function lw = wngpos1(d,l,W,M,lcg,alt,bw,Sw)
%  WNGPOS1 Calculates the location for the leading edge of a triangular
%     wing in a tail controlled missile so that the moment is zero.
%
%           lw = wngpos1(d,l,W,M,lcg,alt,bw,Sw)
%
%           lw    = distance from nose to leading edge of wing (FT)
%           d     = missile diameter (FT)
%           l     = length of missile (FT)
%           W     = missile weight (LB)
%           M     = Mach number
%           lcg   = distance from nose to center of gravity (FT)
%           alt   = altitude (FT)
%           bw    = wing span (FT)
%           Sw    = wing area (FT^2)

Aw = bw^2/Sw;
crw = 2*Sw/bw;
m = bw/2/crw;
sw = 0.5*(d+bw);
beta = sqrt(abs(M^2-1));

Kbw = bigkbwa(M,Aw,d/2,sw);
Kwb = bigkwb(d/2,sw);
CLaW = clawsup(M,Aw);
Sref = pi*(d/2)^2;

%     Calculate the center of pressure of the nose

ln = 2*d;

cpn = cpogvemp(d,ln,M);

XN = lcg-cpn;

ln = 2*d;
Ap = (l-ln)*d+areapogv(d,ln);

q = 0.5*atmsdens(alt)*M^2*atmssofs(alt)^2;

alpha1 = W/q/(2*k2mk1(l/d)*Sref+(Kbw+Kwb)*CLaW*Sw);

cdc = sscfdccc(alpha1,M);

C = [cdc*Ap (2*k2mk1(l/d)*Sref+(Kbw+Kwb)*CLaW*Sw) -W/q];

alpha = max(roots(C));

CLaN = 2*k2mk1(l/d)+cdc*Ap*alpha/Sref;

xcrwb = xcrwba(d/2,sw);

F = beta*Aw*(1+1/m/beta);

if F >= 4
      xcrbw = xcrbwabh(M,d/2,crw);
else
      xcrbw = xcrbwabl(M,Aw,d/2,sw);
```

```
end
lw = ((CLaN*XN*Sref)/(CLaW*Sw)-(Kwb*xcrwb+Kbw*xcrbw)*crw)/(Kwb+Kbw)+lcg;
```

WNGSZEX1

```
function wngszex1()
%  WNGSZEX1 generates a plot using WNGSZTC1 to demonstrates the
%     relationship between wing size and Mach number for varying leading
%     edge sweep angles for a medium range air-to-air missile.  Data
%     used is AMRAAM data from AAMDUS.

d = 0.6;
l = 12;
W = 339;
n = 31;
alt = 10000;
SWP = [45 50 55];
M = linspace(1.2,2,25);
Sw = zeros(3,length(M));
[j1,k1] = size(Sw);

for j = 1:j1

      for k =1:k1

            [bw,Sw(j,k),amax]=wngsztc1(d,l,W,M(k),n,alt,SWP(j));
      end
end

clf
plot(M,Sw)
title('Wing size versus Mach number for varying leading edge sweep
angles')
xlabel('Mach Number')
ylabel('Wing Size (FT^2)')
text(1.3,2.5,'45 DEG')
text(1.3,3.7,'55 DEG')
text(1.33,3.1,'50')
```

WNGSZTC1

```
function
[bw,Sw,amax]=wngsztc1(d,l,W,M,n,alt,SWPwing,amax,lw0,bt0,St0,M2)
%  WNGSZTC1 Is the first approximation of the wing size required for a
%     supersonic, tail-controlled missile.
%
%     [bw,Sw,amax]=wngsztc1(d,l,W,M,n,alt,SWPwing,amax,lw,bt,St,M2)
%
%           bw      = wing span (ft)
%           Sw      = wing surface area (ft^2)
%           amax    = maximum angle of attack (DEG)
%           d       = maximum missile diameter (ft)
%           l       = length of the missile (ft)
%           W       = weight of the missile (lb)
%           n       = load factor (g's)
%           alt     = altitude (ft)
%           SWPwing= leading edge sweep angle (deg)
%           amax    = maximum angle of attack (deg).  amax = is assumed
%                     to be 15 degrees unless another value (~= 0) is
%                     entered.
%           lw      = distance from nose tip to wing leading edge and
%                     body intersection (FT).
%           bt      = tail span (FT)
%           St      = tail size (FT^2) The tail size is assumed to be
%                     0.4*Sw with the same leading edge sweep angle as
%                     the wing unless a value (~= 0) is entered.
%           M2     = load factor of the wing at second Mach number (>1).
%
%     If amax returned is less than amax assumed or entered, then the
%     nose + body provides sufficient lift for the load factor without
%     a wing.
%
%     By placing a zero as default values for amax, bt and St, the M2
%     load factor is calculated with default values.

if nargin < 8
      amax = 15*pi/180;
else
      amax = amax*pi/180;
end

if amax == 0
      amax = 15*pi/180;
end

if nargin < 9
      bt0 = 0;
      St0 = 0;
      lw0 = 0;
end

if nargin < 12
      M2 = 0;
end

amax2 = amax;

if M < 1.2
      M = 1.2;
      disp('Mach number less than 1.2.  Calculations made at Mach 1.2')
```

```
elseif M > 5
      M = 5;
      disp('Mach number greater then 5.  Calculations made at Mach 5')
end

Sref = pi*(d/2)^2;
SWPwing = SWPwing*pi/180;
ln = 2*d;
Ap = d*(l-ln)+areapogv(d,ln);
betaw = sqrt(abs(M^2-1));
Mt = 0.95*M;
betat = sqrt(abs(Mt^2-1));
q = 0.5*atmsdens(alt)*M^2*atmssofs(alt)^2;
nt = (Mt/M)^2;
cdc = sscfdccc(amax,M);

C = [cdc*Ap 2*k2mk1(l/d)*Sref -n*W/q];
maxa = max(roots(C));

if maxa < amax
      amax = maxa*180/pi;
      Sw = 0;
      bw = 0;
else
      %  Calculate the lift curve slope of the nose/body
      CLaN = 2*k2mk1(l/d)+cdc*Ap*amax/Sref;

      %  Calculate the required lift curve slope of the missile
      CLa = n*W/q/amax/Sref;

      %  Calculate the initial wing and tail surface areas
      Sw = W/90;
      dSw = 100000;

      while dSw > 0.0001

            %     Calculate wing parameters

            bw = sqrt(4*Sw/tan(SWPwing));
            sw = 0.5*(d+bw);
            Aw = bw^2/Sw;
            Kbw = bigkbwa(M,Aw,d/2,sw);
            Kwb = bigkwb(d/2,sw);
            CLaW = clawsup(M,Aw);

            %     Calculate tail parameters

            if bt0 == 0|St0 == 0
                  St = 0.4*Sw;
                  bt = sqrt(4*St/tan(SWPwing));
            else
                  St = St0;
                  bt = bt0;
            end

            st = 0.5*(d+bt);
            At = bt^2/St;

            if lw0 == 0
                  deda = 0.5;
            else
```

```
                crw = 2*Sw/bw;
                crt = 2*St/bt;
                lt = 1-crt;
                x0 = (lt-(lw0+crw))/crw;
                deda = Kwb*dedasup(M,SWPwing,x0);

            end

            Kbt = bigkbwna(Mt,At,d/2,st);
            Ktb = bigkwb(d/2,st);
            CLaT = clawsup(Mt,At);
            Sw1 = ((CLa-CLaN)*Sref-(Kbt+Ktb)*nt*CLaT*(1
                                -deda)*St)/CLaW/(Kbw+Kwb);
            dSw = abs(Sw1-Sw);

            Sw = real(Sw1);

        end

        amax = amax*180/pi;

end

if M2 > 1

        cdc2 = sscfdccc(amax2,M2);
        CLaN2 = 2*k2mk1(1/d)+cdc2*Ap*amax2/Sref;

        Kbw2 = bigkbwa(M2,Aw,d/2,sw);
        Kwb2 = bigkwb(d/2,sw);
        CLaW2 = clawsup(M2,Aw);

        Mt2 = 0.95*M2;
        nt2 = (Mt2/M2)^2;

        Kbt2 = bigkbwna(Mt2,At,d/2,st);
        Ktb2 = bigkwb(d/2,st);
        CLaT2 = clawsup(Mt2,At);

        q2 = 0.5*atmsdens(alt)*M2^2*atmssofs(alt)^2;

        n2 = amax2*q2*(CLaN2*Sref+(Kbw2+Kwb2)*CLaW2*Sw
                        +(Kbt2+Ktb2)*CLaT2*nt2*St*(1-deda))/W;
        disp('load factor at M2:')
        disp(n2)
end
```

WVDRGOGV

```
function cdw = wvdrgogv(diameter,length,Mach)
%  WVDROGV Returns the supersonic wave drag for a tangent ogive derived
%     by empirical method of Miles.
%
%           cdw = wvdrgogv(diameter,length,Mach)
%
%           diameter= ogive base diameter
%           length      = length of ogive
%           Mach  = Mach number


sigma = 2*(180/pi)*atan(diameter/2/length);

P = (0.083+0.096/Mach^2)*(sigma/10)^1.69;

lod2 = (length/diameter)^2;

cdw = P*(1-(392*lod2-32)/28/lod2/(Mach+18));
```

XCGFND1

```
function xcg = xcgfnd1(xcgcomp,weight)
%XCGFND1 Calculates the location of the center of gravity for a 1-D
%     system.
%
%           xcg = xcgfnd1(xcgcomp,weight)
%
%           xcgcomp is the center of gravity for each component
%           weight is the weight of the individual component


xcg = sum(xcgcomp*weight')/sum(weight);
```

XCGTC1

```
function xcg=xcgtc1(gclen,gcwt,whlen,whwt,plen,pwt,wfwt)
%  XCGTC1 Calculates the center of gravity of a tail controlled missile
%     from the component data input.  The location of the wing center of
%     is gravity is assumed to be at the finless body center of gravity
%     and the tail center of gravity is assumed to be located at the
%     very end of the missile.
%
%           xcg = xcgtc1(gclen,gcwt,whlen,whwt,plen,pwt,wfwt)
%
%           xcg   = center of gravity
%           gclen = guidance/control subsection length
%           gcwt  = guidance/control subsection weight
%           whlen = warhead subsection weight
%           whwt  = warhead subsection weight
%           plen  = propulsion subsection length
%           pwt   = propulsion subsection weight
%           wfwt  = wing/fin weight

%     Find the center of gravity of the missile without the wing/fins
%     weight or location known.

xcgcomp = [gclen/2 (gclen+whlen/2) (gclen+whlen+plen/2)];
weight = [gcwt whwt pwt];

xcgb = xcgfnd1(xcgcomp,weight);

%     Weight of the tail is 0.4 weight of the wing.
wingwt = wfwt/1.4;
tailwt = 0.4*wingwt;

%     The wing cg is located approximately at the xcgb.
wingcg = xcgb;

%     The tail cg is located all the way at the end of the missile
tailcg = gclen+whlen+plen;

%     Recalculate cg with the wing and tail components added
xcgcomp = [xcgcomp wingcg tailcg];
weight = [weight wingwt tailwt];

xcg = xcgfnd1(xcgcomp,weight);
```

XCGTC2

```
function
xcg=xcgtc2(gclen,gcwt,whlen,whwt,plen,pwt,wngwt,wngcg,tlwt,tlcg)
%  XCGTC2 Calculates the center of gravity of a tail controlled missile
%     from the component data input.
%
%     xcg = xcgtc2(gclen,gcwt,whlen,whwt,plen,pwt,wngwt,wngcg,tlwt,tlcg)
%
%           xcg   = center of gravity
%           gclen = guidance/control subsection length
%           gcwt  = guidance/control subsection weight
%           whlen = warhead subsection weight
%           whwt  = warhead subsection weight
%           plen  = propulsion subsection length
%           pwt   = propulsion subsection weight
%           wngwt = wing weight
%           wngcg = distance of wing center of gravity from nose
%           tlwt  = tail weight
%           tlcg  = distance of tail center of gravity from nose

%     Find the center of gravity of the missile

xcgcomp = [gclen/2 (gclen+whlen/2) (gclen+whlen+plen/2) wngcg tlcg];
weight = [gcwt whwt pwt wngwt tlwt];

xcg = xcgfnd1(xcgcomp,weight);
```

XCRBWABH

```
function xcr = xcrbwabh(M,r,cr)
%  XCRBWABH  Calculates (x/cr)b(w) at super sonic speeds when b*A >= 0
%     (high aspect ratio) and afterbody.
%
%                  xcr = xcrbwabh(M,r,cr)
%
%                  M   = Mach Number
%                  r   = body radius
%                  cr  = root chord

b = sqrt(abs(M.^2-1));

t = 2*b.*r./cr;

p = [1.7611e-2 -1.0474e-1 5.9054e-1 5.0298e-1];

xcr = polyval(p,t);
```

XCRBWABL

```
function xcr = xcrbwabl(M,A,r,s)
%  XCRBWABL  Calculates (x/cr)b(w) at super sonic speeds when b*A < 0
%     (low aspect ratio) and afterbody with no trailing-edge sweep and
%     taper ratio = 0.
%
%                  xcr = xcrbwabl(M,A,r,s)
%
%                  M = Mach Number
%                  A = Aspect ratio
%                  r = Body radius
%                  s = Semispan + r

%            David A. Ekker

b = sqrt(abs(M.^2-1));

p = [1.7976e+0 -8.2857e-1 4.3667e-1 -5.7381e-17];

t1 = b.*A;

t2 = r./s;

slope = polyval(p,t2);

xcr = slope.*t1 + 0.5;
```

XCRBWNAB

```
function xcr = xcrbwnab(M,r,cr)
%  XCRBWNAB  Calculates (x/cr)b(w) at supersonic speeds, b*A >= 0 and no
%     afterbody.
%
%
%                  xcr = xcrbwnab(M,r,cr)
%
%                  M   = Mach Number
%                  r   = body radius
%                  cr  = root chord

b = sqrt(abs(M.^2-1));

t = 2*b.*r./cr;

p = [8.3600e+0 -3.0112e+1 4.3283e+1 -3.1734e+1 1.2746e+1 -3.0908e+0
7.1424e-1 5.0019e-1];

[m,n] = size(t);

xcr = zeros(m,n);

l = m*n;

for k=1:l

      if t(k) > 0.9

            xcr(k) = 2/3;

      else

            xcr(k) = polyval(p,t(k));

      end

end
```

XCRBWSUB

```
function xcr = xcrbwsub(M,A,r,s)
%  XCRWBSUB Calculates the center of pressure for a body in the presence
%     of a triangular in a subsonic flow.
%
%           xcr = xcrwbsub(M,A,r,s)
%
%           xcr   = center of pressure in root chords
%           M     = Mach number
%           A     = aspect ratio
%           r     = body radius
%           s     = r + exposed semispan

b = sqrt(abs(1-M^2));
t = r/s;

if b*A < 4
      p6 = [-1.503003360588808e-004;1.500533636756856e-003;
            -5.843498893035511e-003;1.266301279846288e-002;
            4.997093223254211e-001];
      rs6 = polyval(p6,b*A);

      p4 = [ 1.439562405737517e-004;-2.577574370693720e-003;
             1.915260292425828e-002;-7.636557567970656e-002;
             1.764081283046741e-001;-2.379409374985050e-001;
             1.805435508411277e-001;-7.176958947911805e-002;
             5.130816740444732e-003; 4.999682326019724e-001];
      rs4 = polyval(p4,b*A);

      p2 = [1.870466827291419e-004;-3.055345401484121e-003;
            2.012956338935916e-002;-6.808482446989989e-002;
            1.241295486321807e-001;-1.160715264335058e-001;
            4.940441636896099e-002;-4.087742141651518e-002;
            5.000384633792437e-001];
      rs2 = polyval(p2,b*A);

else
      rs6 = 0.5+2.6*0.05/9;

      rs4 = 0.45+4*0.05/9;

      rs2 = 0.4+1.5*0.05/9;
end

if b*A >= 7
      rs0 = 0.25;
else
      p0 = [ 1.368278964039268e-005;-2.870181449710582e-004;
             2.101837326902790e-003;-6.106506978595964e-003;
             1.093840736149155e-002;-7.485137721564938e-002;
             5.003830336176011e-001];
      rs0 = polyval(p0,b*A);
end

if t > 0.6
      xcr = rs6+(rs6-rs4)*(t-0.6)/0.2;
elseif t > 0.4
      xcr = rs4+(rs6-rs4)*(t-0.4)/0.2;
elseif t > 0.2
```

```
      xcr = rs2+(rs4-rs2)*(t-0.2)/0.2;
else
      xcr = rs0+(rs2-rs0)*t/0.2;
end
```

XCRW

```
function xcr = xcrw(M,A)
%  XCRW  Calculates the wing center of pressure for Mach Number > 1 and
%     taper ratio = 0.
%
%                  xcr = xcrw(M,A)
%
%                  M = Mach Number
%                  A = Aspect Ratio


b = sqrt(abs(M.^2-1));

mb = b.*A;

xcr = (2/3)*ones(size(mb));
```

XCRWBA

```
function xcr = xcrwba(r,s)
%  XCRWBA  Calculates the center of pressure of a triangular wing in the
%     presence of an infinite cylindrical body as given by slender-body
%     theory.
%
%                  xcr = xcrwba(r,s)
%
%                  r = body radius
%                  s = semispan + r

%           David A. Ekker

t = r./s;

p = [-0.0822 0.2009 -0.1190 0.6667];

xcr = polyval(p,t);
```

XCRWBD

```
function xcr = xcrwbd(r,s)
%  XCRWBD  Calculates the center of pressure of lift acting on a
%     triangular wing panel in the presence of a body, from slender body
%     theory.
%
%           xcr = xcrwbd(r,s)
%
%           r = body radius
%           s = maximum semispan + r
%
%     XCRWBD is valid for taper ratio = 0 and trailing edge sweep
%     angle = 0.

%     David A. Ekker

t = r./s;

p = [0.8716 -3.0894 4.3106 -3.1344 1.3829 -0.3885 0.0473 0.6669];

xcr = polyval(p,t);
```

XCRWBSUB

```
function xcr = xcrwbsub(M,A)
%  XCRWBSUB Calculates the center of pressure for a triangular wing in
%     the presence of a body in a subsonic flow.
%
%           xcr = xcrwbsub(M,A)
%
%           xcr    = center of pressure in root chords
%           M      = Mach number
%           A      = aspect ratio

b = sqrt(abs(1-M^2));

if b*A < 2
      p=[-1.130484330484368e-002;5.420357420357615e-002;
         -1.073193473193506e-001;1.446257446257462e-001;
         -1.627277907277905e-001;6.555089355089354e-001];
      xcr = polyval(p,b*A);
elseif b*A >= 5
      xcr = 0.5+3.5/90;
else
      p=[-1.785830762908004e-017;-5.555555555555476e-003;
          5.666666666666665e-001];
      xcr = polyval(p,b*A);
end
```

LIST OF REFERENCES

Beyer, W. H., editor, *CRC Standard Math Tables,* 26th edition, CRC Press, Inc., Boca Raton, FL, 1981.

Bonney, A. E., *Engineering Supersonic Aerodynamics,* McGraw-Hill Book Co., Inc., New York, NY, 1950.

Briggs, M. M., "Systematic Tactical Missile Design", *Tactical Missile Aerodynamics: General Topics,* American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1992.

Bruns, K. D., Moore, M. E., Stoy, S. L., Vukelich, S. R., and Blake, W. B., 'Missile Datcom', AFWAL-TR-91-3039, April 1991.

Chin, S. S., *Missile Configuration Design,* McGraw-Hill Book Co., Inc., New York, NY, 1961.

Devore, J. L., *Probability and Statistics for Engineering and the Sciences,* Third Edition, Brooks/Cole Publishing Company, Pacific Grove, CA, 1991.

Ekin, B., *Dynamics of Atmospheric Flight,* John Wiley and Sons, Inc., New York, NY, 1972.

Fleeman, E. L., "Design of Smart Submunitions", notes from Smart Munitions Technology Course presented to Army Research, Development and Engineering Center (ARDEC), August 1992.

Gerald, C. F., and Wheatley, P. O., *Applied Numerical Analysis,* Fifth Edition, Addison-Wesley Publishing Company Co., Reading, MS, 1994.

Hindes, J. W., "Advanced Design of Aerodynamic Missiles", J. W. Hindes, Goleta, CA, October 1993.

Hoak, D. E., et al, *USAF Stability and Control Datcom,* AFWAL TR-83-3048, Global Engineering Documents, Irvine, CA, 1978.

Hogg, A. P., Tanis, E. A., *Probability and Statistical Inference,* Macmillan Publishing Co., Inc., New York, NY, 1977.

Jerger, J.J., *Principles of Guided Missile Design,* D. Van Nostrand Co., Inc., Princeton, NJ, 1960.

Krieger, R. J. and Vukelich, S. R., "Tactical Missile Drag", *Tactical Missile Aerodynamics: Prediction Methodology,* American Institute of Aeronautics and Astronautics, Inc., Washington DC, 1991.

Lindsey, G.H. and Redmon, D. R., "Tactical Missile Design", notes for missile design class taught at Naval Postgradute School, Monterey, CA, 1980.

Little, J. and Moler, C., *MATLAB User's Guide,* The MathWorks, Inc., Natick, MS, 1993.

Netzer, D., "Tactical Missile Propulsion", lecture notes from missile propulsion class taught at Naval Postgraduate school, Monterey, CA, 1993.

Nicolai, L. M., *Fundamentals of Aircraft Design,* METS, Inc., San Jose, CA, 1984.

Nielsen, J. N., *Missile Aerodynamics,* McGraw-Hill Book Co., Inc., New York, NY, 1960.

Nowell, J. B., *Missile Total and Subsection Weight and Size Estimation Equations,* Master's Thesis, Naval Postgraduate School, Monterey, CA, June 1992.

Ogata, K., *Modern Control Enginnering,* Prentice-Hall Inc., Englewood Cliffs, NJ, 1990.

Pitts, W. C., Nielsen, J.N., and Kaattari, G. E., "Lift and Center of Pressure of Wing-Body-Tail Combinations at Subsonic, Transonic, and Supersonic Speeds", *NACA Tech. Rept. 1307,* 1953.

Sanders, K. L., "An Empirical Method for the Estimation of Downwash", *Ryan Rpt. No. 29254-2A,* 1967.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 52 2
   Naval Postgraduate School
   Monterey, California 93943-5101

3. Dr. Conrad F. Newberry 5
   Dept. of Aeronautics and Astronautics
   AA/Ne
   Naval Postgraduate School
   Monterey, California 93943

4. Dr. Daniel J. Collins 1
   Dept. of Aeronautics and Astronautics
   AA/Co
   Naval Postgraduate School
   Monterey, California 93943

5. Dr. Richard M. Howard 1
   Dept. of Aeronautics and Astronautics
   AA/Ho
   Naval Postgraduate School
   Monterey, California 93943

6. Mr. Emil J. Eichblatt 1
   6351 Palomino Circle
   Somis, California 93066

7. Mr. Eugene L. Fleeman 1
   Program Manager
   Rockwell International Corporation
   Department 394 NC6
   1800 Satellite Boulevard
   Duluth, Georgia 30136

8. Mr. John McCabe 1
Manager, Advanced Design
Northrop Corporation
Aircraft Division
N500/85
One Northrop Avenue
Hawthorne, California 90250-3277

9. Dr. Charles T. Nardo 1
Vice President & General Manager
Aerotherm Corporation
P.O. Box 7040
Mountain View, California 94039-7040

10. Mr. Mick L. Blackledge 1
Assistant Director
SDIO/TNS
Washington, District of Columbia 20301-7100