Report Submitted

to

NASA-Johnson Space Center

An Analytic Model for Footprint Dispersions and its Application to Mission Design

by

J. R. Jagannatha Rao

Assistant Professor

and

Yi-Chao Chen

Assistant Professor

Department of Mechanical Engineering

University of Houston

Houston, TX 77204-4792

Email: rao@tree.egr.uh.edu

November 3, 1992

NASA CR-185706

Report Submitted

to

NASA-Johnson Space Center

An Analytic Model for Footprint Dispersions and its Application to Mission Design

by

J. R. Jagannatha Rao

Assistant Professor

and

Yi-Chao Chen

Assistant Professor

Department of Mechanical Engineering

University of Houston

Houston, TX 77204-4792

Email: rao@tree.egr.uh.edu

November 3, 1992

Summary

This is the final report on our recent research activities that are complementary to those conducted by our colleagues, Professor Farrokh Mistree and students, in the context of Taguchi method. We have studied the mathematical model that forms the basis of the Simulation and Optimization of Rocket Trajectories (SORT) program and developed an analytic method for determining mission reliability with a reduced number of flight simulations. This method can be incorporated in a design algorithm to mathematically optimize different performance measures of a mission, thus leading to a robust and easy-to-use methodology for mission planning and design.

1 Introduction

NASA currently employs an entry Monte Carlo analysis incorporated with the Simulation and Optimization of Rocket Trajectories (SORT) program to analyze trajectories of space vehicles. A basic function of this analysis is to assess the dispersion of a trajectory from the prescribed one due to dispersions of various vehicle and environmental parameters.

As an example, the Monte Carlo analysis of the LifeSat vehicle has been performed to determine:

1. If a trim burn is needed to correct state vector after de-orbit burn.

2. The expected G-load necessary to land within desired target area.

Three scenarios of state vector correction after de-orbit burn were investigated. In each scenario, five sets of vehicle and environmental parameters were dispersed: initial state, atmosphere data, winds, vehicle and parachute aerodynamics, and vehicle weight. Also, five entry interface conditions characterized by peak G-load were analyzed for each of the second and third scenarios. To achieve 95% confidence of 99.73% reliability, 1109 dispersed cases were needed for each G-load.

Based on these simulations, footprint distributions were plotted for all cases of G-load, and footprint dispersions computed as functions of G-load for the given 99.73% reliability. It was then concluded that a trim burn with a 14g to 15g entry is required to land within the desired target area. All parameters were recorded for statistical post-processing.

While the Monte Carlo analysis generates satisfactory results, the necessary confidence with this method is supported by large numbers of simulations. The associated high costs not only make a design process expensive, but also severely limit the number of scenarios

that can be considered in the design.

It is then desirable to develop alternative methods that would significantly reduce the number of simulations. In a broad sense, different methods can be classified into two categories. One is based on a statistical approach, and the other on an analytic approach. A statistical method uses a collection of samples to simulate a stochastic process, while an analytic method models the process through probability density functions.

The analytic method we have developed makes use of the fact that the desired landing area is small in an appropriate scale, which make it possible to approximate admissible footprint dispersions as simple functions of the dispersions of the vehicle and environmental parameters. These functions can be determined by a small number of simulations. The set of admissible parameters then can be defined through certain functional relations, and the reliability associated with given parameter distributions can be evaluated by a high dimensional integral. This integral can be further reduced, by an appropriate coordinate transformation, to a two-dimensional one, that can be readily evaluated numerically.

2 Statement of the Problem

A returning space vehicle is to land within a desired target area. Once the vehicle crosses the entry interface, its landing position (footprint) depends only on a number of vehicle and environmental parameters, including initial position and velocity of the vehicle at the entry interface, atmospheric density, pressure, temperature, wind speeds, wind direction, angle of attack, drag and lift coefficients of the vehicle and parachutes, reference areas of the parachutes, and vehicle mass. If all parameters take their nominal values, the vehicle

will follow a predetermined trajectory and land at the center of the target area. In a real flight, deviations of the parameters from the nominal values will inevitably occur, resulting in a deviation of the footprint from the center.

Due to their stochastic nature, the values of the parameters cannot be predicted exactly. They can take any value in certain ranges. However, their variations obey specific statistical descriptions, that can be expressed either by discrete data drawn from a large number of samples, or by continuous probability density functions determined by experiments.

A consequence of such statistical distributions of the parameters is that the deviation of the footprint also obeys certain statistical rules, making it possible to define the reliability of a mission as a measure of likelihoods of having an admissible landing. Precisely, given the distributions of the vehicle and environmental parameters, the reliability P of vehicle landing is defined as the probability for the vehicle to land within the desired target area. To determine P, two different approaches are possible as described below.

3 Statistical and Analytic Approaches

By its physical interpretation, the probability of an event is the limit of the relative frequency of occurrence of the event in an infinite sequence of replications. It is then not unreasonable to take the relative frequency in a finite sequence as an approximation of the probability. This is the basic idea that the statistical approach is based upon.

In the Monte Carlo analysis, a number of experiments (computer simulations of vehicle trajectories) are carried out, each based on a randomly chosen set of parameter values. If the distribution of a parameter is given by a set of discrete data, a data point is taken randomly

from the set in an experiment. On the other hand, if the distribution is given by a probability density function, a data point is generated randomly in accordance with the given function relation in a statistical sense. The footprints of all experiments are calculated, and those within the target area identified. The reliability is then given approximately by

$$P \cong \frac{\text{number of admissible landing}}{\text{number of experiments}}. \tag{1}$$

The error due to the approximation in Eq.(1) depends on the number of experiments. It is usually measured in terms of confidence, which is defined as the probability of correct prediction. As stated in Section 1, some 1109 experiments are required to achieve 95% confidence. The large number of experiments associated with this approach is a major source of high costs, and appears unavoidable due to the high confidence requirement.

Other methods within the statistical approach have been proposed. For instance, by a Taguchi method, experiments are performed on a selected set of data points, that are generated by using orthogonal arrays. As the Taguchi method has proved capable of delivering useful information on the distribution of footprints, an analysis of reliability based on this method has not yet been developed.

The basic idea of the analytic approach is to describe the distributions of the parameters by probability density functions, and to compute the reliability directly by the mathematical definition of probability rather than by running statistical experiments. Computer simulations are needed only for determining admissible parameters, not for establishing statistical profiles of footprints. This provides the possibility of significant savings on simulation count. Also, this approach completely eliminates the issue of confidence as the result is not obtained

by using a finite sequence to approximate an infinite sequence, and therefore has no error from that source.

The main difficulty associated with the analytic approach is related to its numerical feasibility. For a system having as many parameters as a space vehicle does, the determination of admissible parameters is a tedious and laborious process, if possible to accomplish at all. The method we developed uses an approximation scheme to simplify this process, and consequently leads to the computation of reliability in a concise way.

The method is presented in the following sections.

4 Mathematical Formulation

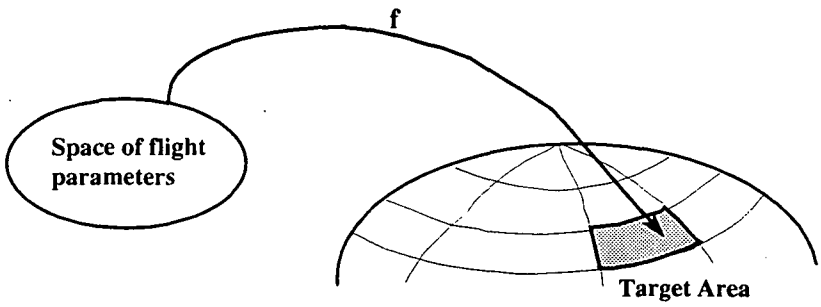

Figure 1: The footprint mapping f.

Suppose that there are n vehicle and environmental parameters $x_1, x_2, \ldots, x_n$, that determine the trajectory and therefore the footprint of the vehicle. Let $\mathbf{f}$ be the 2-dimensional vector whose components are the coordinates, longitude and geodetic latitude for example, of the footprint. Then we can write

$$\mathbf{f} = \mathbf{f}(x_1, x_2, \ldots, x_n),$$

where we also use $\mathbf{f}$ to denote a function that maps the set of the parameters to the set of footprints. This function is given numerically by integrating the trajectory equations.

A rectangular target area can be expressed by

$$D = \{(f_1, f_2) : c_1 - d_1 \leq f_1 \leq c_1 + d_1, c_2 - d_2 \leq f_2 \leq c_2 + d_2\} \tag{2}$$

where (c_1, c_2) corresponds to the nominal landing coordinates, and d_1 and d_2 are the allowed longitude and geodetic latitude deviations of footprint. A landing with footprint $\mathbf{f}$ is admissible if $\mathbf{f} \in D$. A set of parameter values $(x_1, x_2, \ldots, x_n)$ is said to be admissible if $\mathbf{f}(x_1, x_2, \ldots, x_n) \in D$. The set of all admissible parameters is denoted by

$$\Omega = \{(x_1, x_2, \ldots, x_n) : \mathbf{f}(x_1, x_2, \ldots, x_n) \in D\}. \tag{3}$$

The distribution of a parameter x_i is assumed to be given by a probability density function $p_i(x_i)$. The value $p_i(a_i)$ corresponds to the probability for x_i to take values in an infinitesimal neighborhood of a_i divided by the length of the neighborhood. The probability for the parameters $(x_1, x_2, \ldots, x_n)$ to take values in a neighborhood of $(a_1, a_2, \ldots, a_n)$ divided by the n-dimensional volume of the neighborhood is then

$$p_1(a_1)p_2(a_2)\ldots p_n(a_n).$$

It is now readily seen that the reliability, or the probability of landing in the target area, is given by

$$P = \int_\Omega p_1(x_1)p_2(x_2)\ldots p_n(x_n)dx_1 \cdots dx_n. \tag{4}$$

While Eq. (4) gives the exact expression of reliability, the evaluation of the integral is difficult due to the presence of two major difficulties. First, the determination of the set Ω of admissible parameter as defined by Eq. (3) involves finding the pre-image of function $\mathbf{f}$ under D. Since the explicit expression of $\mathbf{f}$ is not available, one has to numerically solve two equations of n unknowns variables. In general, this requires extensive numerical iterations. Secondly, the integral Eq. (4) is n-dimensional with the domain of integration Ω given in a numerical form. Since n is usually a large number, fifty say, the numerical integration can be very lengthy. In the next two sections, we shall show how these difficulties can be overcome by appropriate linear approximation and transformation of coordinates.

5 Approximation

Determined by the solutions of nonlinear differential equations, the footprint function $\mathbf{f}$ is expected to be nonlinear. However, since the target area is small in a scale corresponding to the level of the linear momentum and other flight variable of the vehicle, function $\mathbf{f}$ can be approximated by lower order polynomials when restricted to the admissible set Ω.

This fact is clearly demonstrated by Figures 4-6 which show the dependencies of the footprint coordinates on the atmospheric density, the drag coefficient of the vehicle, the wind speed, and the initial velocity for the LifeSat vehicle. It is observed that in the admissible dispersion ranges, the errors induced by linear approximations of $\mathbf{f}$ are less than 8%. We expect similar behavior for the dependences of $\mathbf{f}$ on other vehicle and environmental parameters.

This suggests that we approximate $\mathbf{f}$ by first order polynomials as

$$f_1(x_1, x_2, \ldots, x_n) = c_1 + g_{11}(x_1 - \mu_1) + g_{12}(x_2 - \mu_2) + \ldots + g_{1n}(x_n - \mu_n),$$

$$f_2(x_1, x_2, \ldots, x_n) = c_2 + g_{21}(x_1 - \mu_1) + g_{22}(x_2 - \mu_2) + \ldots + g_{2n}(x_n - \mu_n), \qquad (5)$$

where $\mu_1, \mu_2, \ldots, \mu_n$ are nominal values of the parameters, and $g_{ij}, i = 1,2, j = 1,2,\ldots,n$, are the derivatives of f_i with respect to x_j evaluated at the nominal values. We note that totally $2n$ simulations are needed to determine the approximate footprint functions of form in Eq. (5).

Substituting Eq. (5) into Eq. (2) and Eq. (3), we find that the set Ω of all admissible parameters can be determined approximately by the following inequalities

$$-d_1 \leq g_{11}(x_1 - \mu_1) + g_{12}(x_2 - \mu_2) + \ldots + g_{1n}(x_n - \mu_n) \leq d_1,$$

$$-d_2 \leq g_{21}(x_1 - \mu_1) + g_{22}(x_2 - \mu_2) + \ldots + g_{2n}(x_n - \mu_n) \leq d_2 \qquad (6)$$

Inequalities (6) define a region between two pairs of parallel $(n-1)$-dimensional planes in the n-dimensional parameter space.

6 Integration of Probability Density Functions

Even with the set Ω of admissible parameters in the approximate form of Eq. (6), the evaluation of the n-dimensional integral in Eq. (4) is still not an easy task. The difficulties lie in the fact that the integral limits are not constants, but functions of integral variables.

Also, the high dimension of integration makes the numerical treatment cumbersome.

In practice, the probability density function $p_i(x_i)$ of the parameter x_i often has the form of either a uniform distribution (Figure 2)

$$p_i(x_i) = \begin{cases} \frac{1}{2h} & \text{if } \mu_i - h_i \leq x_i \leq \mu_i + h_i \\ 0 & \text{otherwise} \end{cases}$$

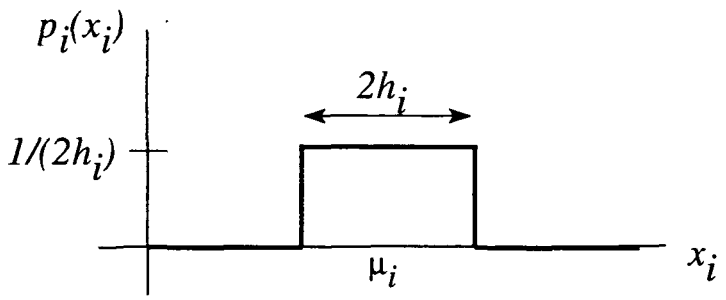

Figure 2: A uniform distribution.

or a normal distribution (Figure 3)

$$p_i(x_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}} \tag{7}$$

where σ_i is the standard deviation. The integration of a uniform distribution is immediate and needs no discussion. In the following, we consider the case where $p_i(x_i)$ in Eq. (4) are of the form in Eq. (7).

The idea is to introduce a transformation of coordinates so that the integral in Eq. (4) can be converted to one with constant limits, and the integration can be carried out explicitly except for two variables. Numerical integration is then performed on the resulting 2-dimensional integral.

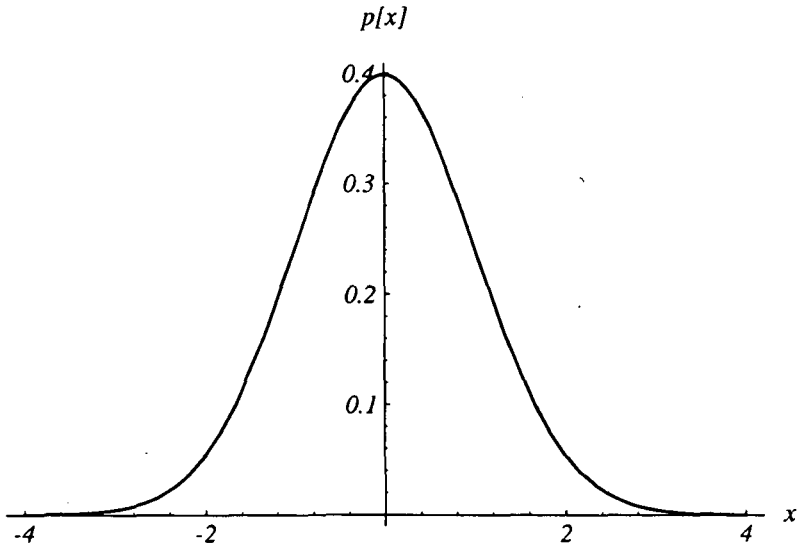

Figure 3: A normal distribution.

To this end, we introduce new variables

$$\xi_1 = g_{11}(x_1 - \mu_1) + g_{12}(x_2 - \mu_2) + \ldots + g_{1n}(x_n - \mu_n),$$

$$\xi_2 = g_{21}(x_1 - \mu_1) + g_{22}(x_2 - \mu_2) + \ldots + g_{2n}(x_n - \mu_n).$$

Taking ξ_1, ξ_2 and $n-2$ of $x_1, x_2, \ldots, x_n$ as integral variables, we can transfer (4) into an integral for which the integral domains of ξ_1 and ξ_2 are $[-d_1, d_1]$ and $[-d_2, d_2]$, respectively, and those of x's are $(-\infty, \infty)$. Furthermore, a detailed analysis shows that the $n-2$ fold integration in x's can be carried out analytically, leading to the following 2-dimensional integral for the reliability

$$P = \frac{1}{2\pi\sqrt{A}} \int_{-d_2}^{d_2} \int_{-d_1}^{d_1} e^{-\frac{G(\xi_1,\xi_2)}{2A}} d\xi_1 d\xi_2, \tag{8}$$

where

$$A = \sum_{1 \le i < j \le n} \sigma_i^2 \sigma_j^2 (g_{1i} g_{2j} - g_{1j} g_{2i})^2,$$

$$G(\xi_1, \xi_2) = \sum_{i=1}^{n} \sigma_i^2 (g_{1i} \xi_2 - g_{2i} \xi_1)^2.$$

7 Simplified LifeSat Model

In this section, we illustrate the method described above by applying it to the simplified LifeSat model constructed by M. A. Tigges. In particular, 3 parameters, atmospheric density $\rho = x_1$, vehicle drag coefficient $C_d = x_2$ and wind speed $w = x_3$, are dispersed, while other parameters are set at their nominal values. The 3 dispersed parameters contribute to the majority of the dispersion of the footprint.

A footprint is described by the longitude f_1 (deg) and the geodetic latitude f_2 (deg). The target landing area is defined by $f_1 = c_1 \pm d_2$ and $f_2 = c_2 \pm d_2$, where

$$c_1 = -106.66,\ c_2 = 33.6,\ d_1 = 0.08,\ d_2 = 0.2.$$

In Figures 4-6, we plot f_1 and f_2 vs. ρ, C_d and w. The ranges of the parameters are chosen according to their dispersion values: 30% for ρ, 5% for C_d, and $\pm$ 30 mph for w. Longitude f_1 is constant in ρ and C_d, as is apparent from the physics. Similarly is geodetic latitude f_2 in w. Other function dependences are non-constant and nonlinear. However, in the dispersion ranges of interest, the errors caused by linear approximations are less than 8%. If quadratic approximations are employed, the errors would be practically negligible. Similar behavior is expected for the dependences of f_1 and f_2 on other parameters.

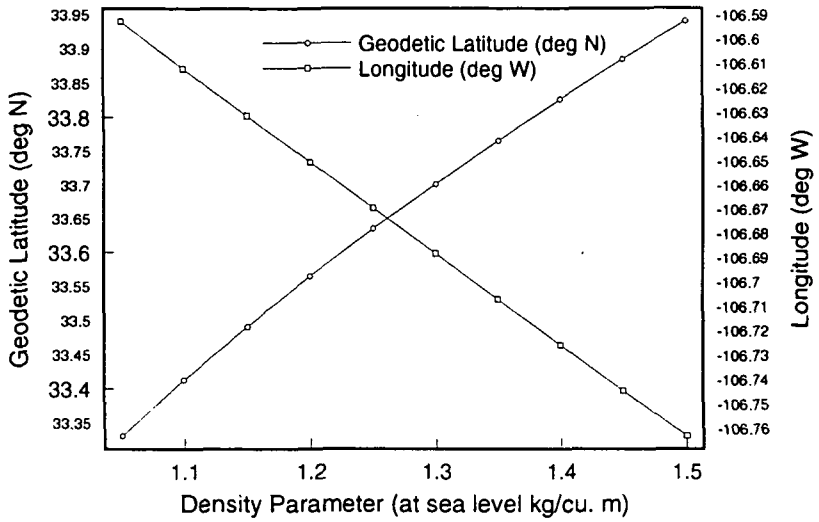

Figure 4: Longitude and geodetic latitude vs. atmospheric density

The derivatives of f_1 and f_2 with respect to ρ, C_d and w for the case of 13 g-load can be readily calculated as

$$g_{11} = \frac{\partial f_1}{\partial \rho} = -0.370,\ g_{12} = \frac{\partial f_1}{\partial C_d} = -0.367,\ g_{13} = \frac{\partial f_1}{\partial w} = 1.47 \times 10^{-3},$$

$$g_{21} = \frac{\partial f_2}{\partial \rho} = 1.24,\ g_{22} = \frac{\partial f_2}{\partial C_d} = 2.50,\ g_{23} = \frac{\partial f_2}{\partial w} = 6.57 \times 10^{-5}$$

Results

The following results were now directly obtained with for the two indicated cases of density dispersions.

Case 1

Disperse ρ normally by 30 % (3 σ)

C_d normally by 5 % (3 σ)

w normally by 17 mph.

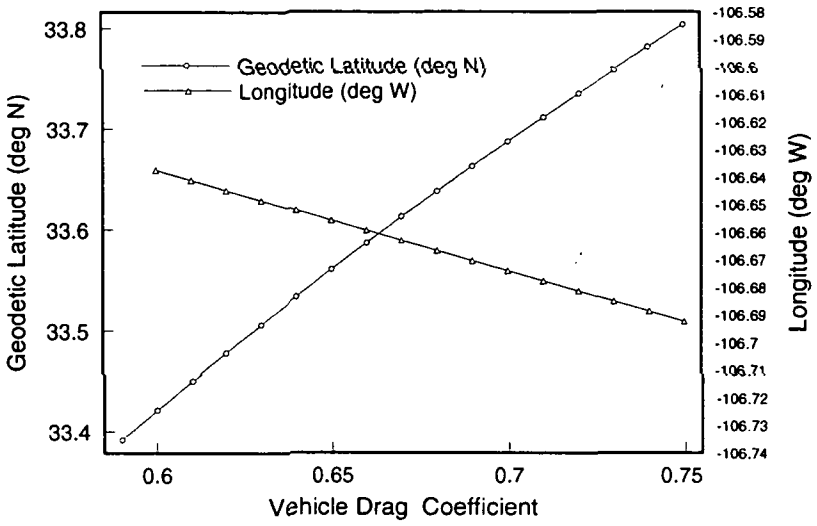

Figure 5: Longitude and geodetic latitude vs. vehicle drag coefficient

Peak G-Load	$12g$	$13g$	$14g$	$15g$
Reliability P(%)	69.91	74.16	79.71	83.73

Case 2

Disperse ρ normally by 15 % (3 σ)

C_d normally by 5 % (3 σ)

w normally by 17 mph.

Peak G-Load	$12g$	$13g$	$14g$	$15g$
Reliability P(%)	95.31	96.96	98.51	99.22

The reliability variations with the maximum G-load during the flight, as shown in Figure 7, are very important during mission planning. **Note that for the 3 dispersed variables in this case, only 6 simulations were required to arrive at the results summarized above.**

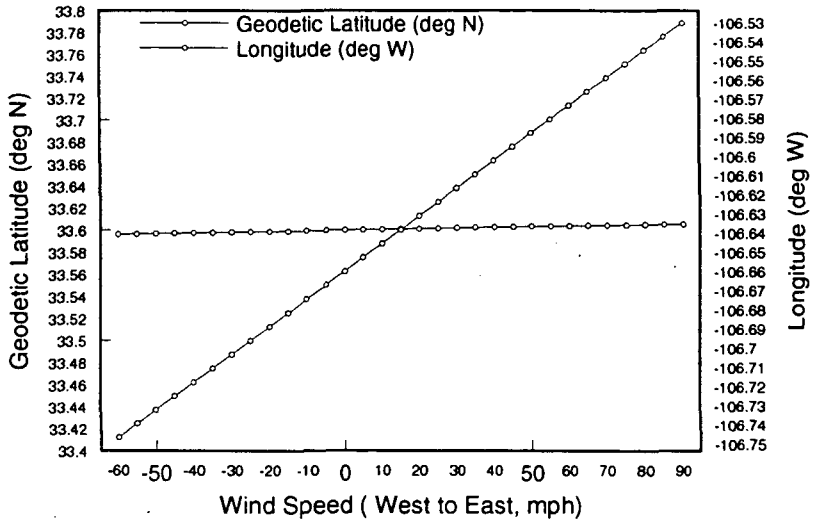

Figure 6: Longitude and geodetic latitude vs. wind speed

Integration into a Design Strategy

As mentioned above, our efficient method for computing reliabilities makes possible the following very desirable scenario - we wish to use reliability predictions during the design and planning of the mission itself (i.e., analysis "concurrently" with design) and not just as a performance analysis tool.

For a typical space mission, two classes of variables affect vehicle performance: (1) the deterministic design variables such as vehicle size and shape, drag and lift coefficients and pay load capacity, and (2) stochastic variables such as atmospheric density, temperature, and wind conditions. The mission designer has direct control over only the design variables whereas the stochastic variables are typically described by their probability density functions. With this natural partition of variables governing the space vehicle, what is a rational design strategy?

Consider, for the entry and landing phase of the LifeSat vehicle, the following different design statements:

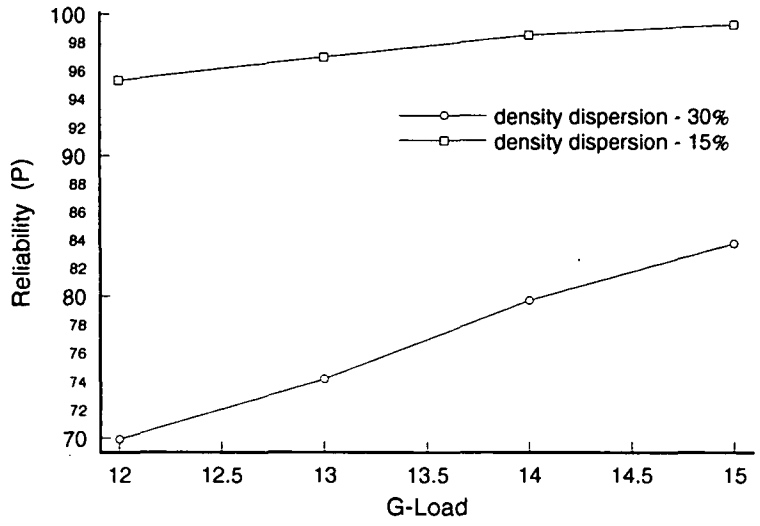

Figure 7: Reliability variations with max. G-load

1. Find the (best) nominal values of the design variables and the entry interface conditions which minimize the target landing error and which satisfy flight constraints (e.g., the g-load not to exceed 12).

2. Design a LifeSat mission which maximizes the reliability of landing within the target and for which the g-load does not exceed 12.

3. Design a LifeSat mission which maximizes the pay load capacity such that the reliability of acceptable landing exceeds 99.73% and the g-load during the flight does not exceed 12.

These and other design models are immediately accessible to us since we have a simplified mathematical model for the flight trajectory and we also have a very efficient method to estimate the measure of reliability. These word statements of design problems can be naturally transcribed into nonlinear programming models for which available computational methods can be used to search for optimum designs. For the LifeSat model, we have already

formulated and solved a few of the above mentioned design problems. For example, for the fixed data given below:

altitude = 0.12192d6	azimuth = 180.0	clift = 0.0	cdrogue-drag = 0.55
cmain-drag=0.8	drogue-area=4.104	cmain-area=51.234	drogue-mach=1.5
cmain-altitude=3048.0	density-sl=1.225	tstep=2.0	wind-speed = 0.0

the following nominal values were obtained for the design variables and EI conditions:

EI Variable	l.b	u.b.	init.	final
latitude (deg N)	40.0	60.0	40.0	44.498
longitude (deg W)	-110.0	-100.0	-110	-105.507
flight path (deg)	-7.0	0.0	-5.0	-6.037
velocity (m/s)	9000	11,000	9800	9799.8
angle of attack (deg)	-5.0	5.0	0.0	0.0
rotational speed (deg/s)	0.0	50.0	25.0	25.0
mass (kg)	1200.0	1700.0	1550	1550.1
vehicle drag	0.6	0.9	0.7	.8427
frontal area (sq. m)	2.0	4.0	3.1	3.6357

An interesting 'what-if' scenario for this model is to determine what is the effect of the g-load constraint on the flight. The following is a comparison of one set of optimal nominal values for the two cases (with and without g-load constraint):

Variable	No Constraint	Max G-Load= 12.0
latitude (deg N)	44.498	52.21
longitude (deg W)	-105.507	-105.327
flight path (deg)	-6.037	-5.021
inertial velocity (m/s)	9799.8	9799.56
angle of attack (deg)	0.0	-5.0
alpha rotational speed (deg/s)	25.0	25.0
satellite mass (kg)	1550.1	1549.8
vehicle drag	0.8427	0.696
vehicle frontal area (sq. m)	3.6357	3.167
final-latitude (deg N)	33.603	33.641
final-longitude (deg W)	-106.659	-106.673
time of flight (sec)	375	485

Figures 8 and 9 show the time history for the two cases.

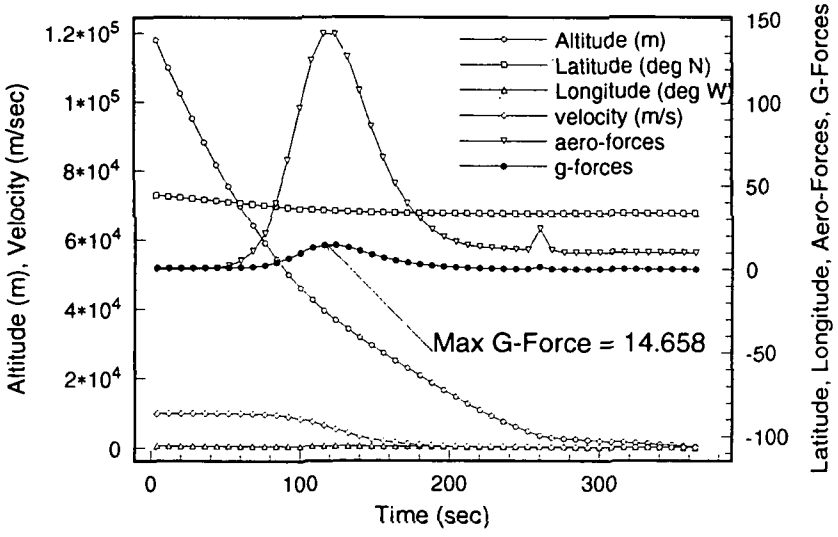

Figure 8: Optimum flight without g-load constraint

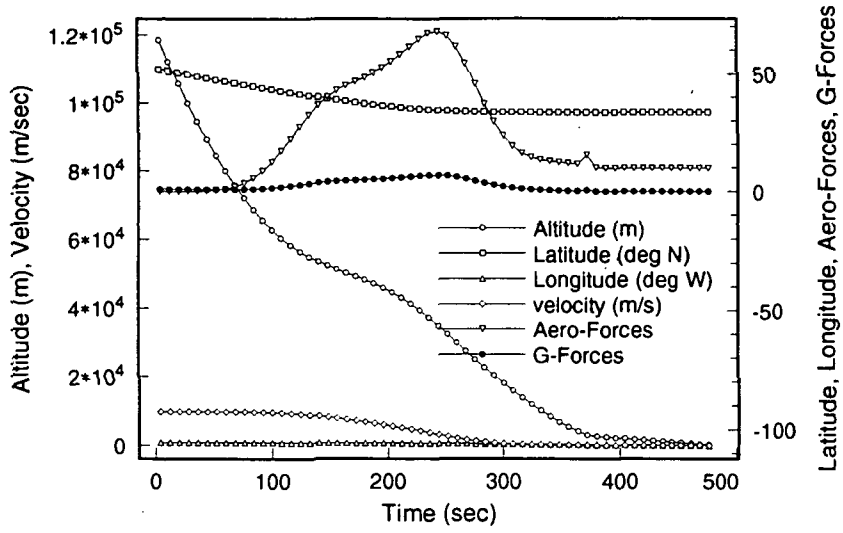

Figure 9: Optimum flight with g-load constraint

Further, Figure 10 shows that there are indeed nominal values of the variables such that the g-load can be made as low as 7.5. Note that there was no constraint on reliability in this model. Thus, further work needs to be done to investigate this design since we know from Figure 7 that a low g-load mission typically has low reliability.

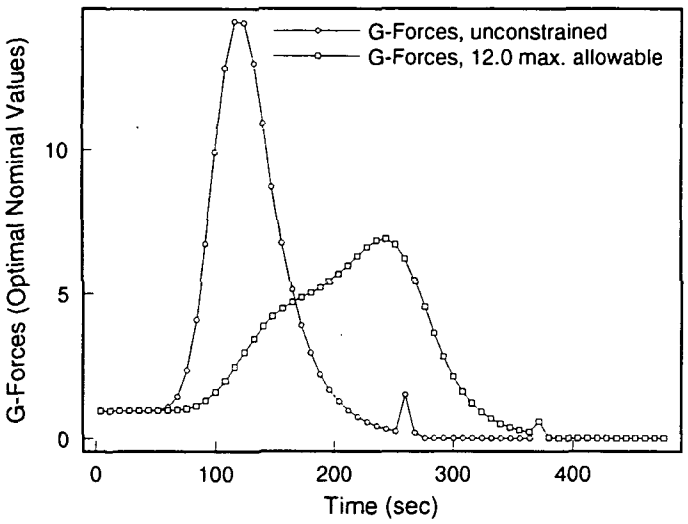

Figure 10: Reliability variations with max. G-load

8 Conclusions and Generalization to other Missions

Further test and development of the analytical model is proposed. Following our project NAG 9-616 with NASA-JSC, an immediate step is to fully implement the model on LifeSat in which we will

- Compare our predictions with Monte Carlo analysis. Assess the error due to the approximation employed.
- Study more accurate approximations, as well as other types of probability density functions.

Also to be investigated is the problem of generalizing the analytical model so that it can be applied to other systems for which the reliability of performance is an important concern, as is the case for many problems in mission control. As our collaboration with the Systems Engineering Division of NASA-JSC continues, we will identify and solve additional problems of importance. In general, the applicability of our analytical model is based upon two basic assumptions: (a) Some simplified form of the governing equations is available, and (b) The probability density functions are explicit. It must be pointed out that the simplifications in the governing equations are not a handicap since this method is being developed as a design tool and not as a replacement for detailed Monte Carlo simulations.

9 Program SATSIM

The following files are necessary to run the program SATSIM: satsim.f, lsoda.f, satsim.dat. The flight data is input via satsim.dat. lsoda.f is a ODE solver from the well-known ODEPACK. The subroutine that includes the machine specific constant in lsoda is also included here for completeness. The rest of the solver routines can be down loaded from NETLIB (ftp to research.att.com). On a unix machine, the following commands are used to compile and run SATSIM:

```
% f77 -c lsoda.f
% f77 -o satsim satsim.f lsoda.o
% satsim
```

File satsim.dat

```
0.12192d6       initial altitude  (m)
45.8017  geodetic latitude (deg) [i.e., 90 - theta]
-105.42323          longitude, west, (deg) [i.e., phi - 180]
-5.7607 flight path angle, (deg)
9946.57         inertial velocity (m/sec) .
180.0  azimuth angle (deg)
********* other problem specific data ***********
1557.66   satellite mass (kg) 3434 lb * .45359
0.0 angle of attack (deg)
0.0  vehicle lift coeff.
0.66512 vehicle drag coeff.
0.55 drogue chute drag coeff.
0.8  main chute drag coeff.
4.8616          vehicle area (sq. m)
4.104  drogue chute area (sq. m)
51.234 main chute area (sq. m)
1.5        drogue chute deployment Mach number
3048.0 main chute deployment altitude (m)
1.225 atm. density at sea level (kg/cu.m )
25.0 alpha rotational speed (deg/sec)
2.0 time step for integration (sec)
0.0        wind speed (mph, positive west to east)
```

File satsim.f

```
c*****************************************************************
c satsim.f version 1
c August 20, 1992
c Earth
c rotation effects added; other flight data is provided for
c for plotting. time step for integration is input in
```

```
c the data file. The only item yet to be added is drogue
c chute deployment. The main chute deploys.
c*****************************************************************

c*****************************************************************
c satsim.f - Version 1
c
c simple simulation program for lifesat vehicle
c June 17, 1992
c  by J. R. Jagannatha Rao
c
c The equations have been derived in spherical coordinates
c and will be integrated using the odepack solver LSODA.
c At this time, the inertial effects due to the rotation of the
c earth have not been included.
c
c*****************************************************************


      implicit real*8(a-h,o-z)

      external fex

c these are parameters required by lsoda and should be
c changed for a new problem.
      parameter (neq=6,lrw=150,liw=30)
      parameter (ndimy = 30)
c input data  file unit number
      parameter (idata = 8)
      parameter (iecho=6)
      parameter (iplot=12)

      dimension y(ndimy), atol(neq), rwork(lrw), iwork(liw)

c open proper files
open(unit=idata, file='satsim.dat')
open(unit=iplot, file='satsim.plt')


pi = 4.0*atan(1.0)
degree = pi/180.0
earth_radius = 6.377656d6  ! (m)
write(iecho,*)
write(iecho,*) ' LANDSTAT SATELLITE SIMULATION PROGRAM '
write(iecho,*) '             Input Data:- '
```

```
c initial values of the state vector
c
c y(1) := r
read(idata,*) altitude
write(iecho,*) 'altitude =>  ', altitude, ' (m)'
      y(1) = altitude + earth_radius
c y(3) := theta
read(idata,*) geod_lat
c234567890123456789012345678901234567890123456789012345678901234567890
write(iecho,*) 'geodetic latitude =>  ', geod_lat, ' (deg)'
      y(3) = (90 - geod_lat)* degree
c y(5) := phi
read(idata,*) wlongitude
write(iecho,*) 'longitude => ', wlongitude, ' (deg. W)'
      y(5) = (180 + wlongitude) * degree

c read flight path angle, (deg)
read(idata,*) flight_path
write(iecho,*) 'flight path => ', flight_path, ' (deg)'
c convert to radians
flight_path = flight_path*degree
c read inertial velocity, vel_init (m/sec)
read(idata,*) vel_init
write(iecho,*) 'inertial velocity => ', vel_init, ' (m/sec)'
c read azimuth angle, (deg)
read(idata,*) azimuth
write(iecho,*) 'azimuth angle =>', azimuth, ' (deg)'
c convert to radians
azimuth = azimuth*degree

c y(2) := rdot
rdot = vel_init*dsin(flight_path)
       y(2) = rdot
c y(4) := thetadot
thetadot = vel_init*dcos(flight_path)/y(1)
       y(4) = thetadot
c y(6) := phidot (from azimuth angle)
       phidot=(vel_init/(y(1)*dsin(y(3))))*dcos(azimuth - pi/2.0)
       y(6) = phidot
c read other problem specific data
y(7) = earth_radius
c read a comment  line
read(idata,*)
c satellite mass, satmass
read(idata,*) satmass
write(iecho,*) 'Satellite Mass => ', satmass, ' (kg)'
```

```
y(8) = satmass
c angle of attack, alpha_init
read(idata,*) alpha_init_deg
       write(iecho,*) 'angle of attack => ', alpha_init_deg,' (deg)'
y(9) = alpha_init_deg*degree
c vehicle lift coefficient,clift
read(idata,*) clift
y(10)=clift
c vehicle drag coefficient, cdrag
read(idata,*) cdrag
y(11)=cdrag
c drogue chute drag coefficient, cdrogue_drag
read(idata,*) cdrogue_drag
y(12) =  cdrogue_drag
c main chute drag coefficient, cmain_drag
read(idata,*) cmain_drag
y(13) = cmain_drag
c vehicle frontal area, varea
read(idata,*) varea
y(14)= varea
c drogue chute area, drogue_area
read(idata,*) drogue_area
y(15)=drogue_area
c main chute area, cmain_area
read(idata,*) cmain_area
y(16)= cmain_area
c drogue  chute deployment Mach number
read(idata,*) drogue_mach
y(17)= drogue_mach
c main chute deployment altitude
read(idata,*) cmain_altitude
y(18) = cmain_altitude
c atm. density at sea level
read(idata,*) density_sl
y(19) = density_sl
c alpha rotational speed
read(idata,*) alpha_rate
y(20) = alpha_rate
c time step for integration
read(idata,*) tstep
c wind speed, mph, positive west to east
read(idata,*) wind_speed
y(21) = wind_speed * 0.44704


c initial value of the independent variable 't'
```

```
      t = 0.0d0

c first point where output is desired
      tout = 1.0d0

c itol is 1 if atol is scalar, 2 if atol is an array
      itol = 2

c relative tolerance parameter (scalar)
      rtol = 1.0d-6

c absolute tolerance parameter(s)
      atol(1) = 1.0d-4
      atol(2) = 1.0d-4
      atol(3) = 1.0d-6
      atol(4) = 1.0d-6
      atol(5) = 1.0d-6
      atol(6) = 1.0d-6
c
      itask = 1
c
      istate = 1
c
      iopt = 0

c use the following if optional inputs are being used..
c       do 22 i=5,10
c       iwork(i) = 0
c       rwork(i)=0.0
c  22   continue
c  iwork(6) = 500

c
      jt = 2

c main integration cycle
do while (y(1) .ge. earth_radius)
        call lsoda(fex,neq,y,t,tout,itol,rtol,atol,itask,istate,
     1     iopt,rwork,lrw,iwork,liw,jdum,jt)
        geod_lat = 90 - (y(3)/degree)
wlongitude = y(5)/degree - 180.0
        write(iecho,20)t,y(1), geod_lat , wlongitude
  20    format(7h at t =,e12.4,6h   y =,3e14.6)
        if ((istate .lt. 0) )  then
  80          write(iecho,90)istate
  90          format(///22h error halt.. istate =,i3)
```

```
              stop
 end if

  40    tout = tout + tstep

       write(iecho,60)iwork(11),iwork(12),iwork(13),iwork(19),rwork(15)
  60   format(/12h no. steps =,i4,11h  no. f-s =,i4,11h  no. j-s =,i4/
     1    19h method last used =,i2,25h   last switch was at t =,e12.4)

c write quantities in the plot file
       call forces (neq,t,y, fr, ftheta, fphi,velnorm,aero_forces,
     1 g_forces)
current_altitude = y(1) - earth_radius
write(iplot,91) t, current_altitude, geod_lat, wlongitude,
     1 velnorm,  aero_forces,g_forces
 91 format(1x,f6.2,1x,f10.2,1x,f10.3,1x,f10.3,1x,f14.5,1x,f8.3,
     1 1x,f8.3)
       end do

write(iecho,*) '******************'
write(iecho,*) ' satellite landed !'
geod_lat = 90 - (y(3)/degree)
wlongitude = y(5)/degree - 180.0
write(iecho,*)' geod_lat =>',geod_lat,
     1  '  longitude =>',wlongitude

       stop
       end

       subroutine fex (neq, t, y, ydot)
       implicit real*8(a-h,o-z)
       dimension y(30), ydot(6)
       r = y(1)
       rdot = y(2)
       theta = y(3)
       thetadot = y(4)
       phi = y(5)
       phidot = y(6)
       earth_radius  = y(7)
       satmass = y(8)
       alpha_init = y(9)
       clift = y(10)
       cdrag = y(11)
       cdrogue_drag = y(12)
       cmain_drag = y(13)
       varea = y(14)
```

```
      drogue_area = y(15)
      cmain_area = y(16)
      drogue_mach = y(17)
      cmain_altitude = y(18)
      density_sl = y(19)
      wind_speed = y(21)
      wind_component = -1.0* wind_speed/(r*dsin(theta))

c account for earth rotation
pi = 4.0*atan(1.0)
degree = pi/180.0
      earth_rot = (1.0/240.0)*degree
      phidot = phidot + earth_rot + wind_component

      call forces (neq,t,y, fr, ftheta, fphi,velnorm,aero_forces,
     1 g_forces)
      ydot(1) = y(2)
      ydot(2) = fr + r*(thetadot**2 + phidot**2 *(dsin(theta)**2))
      ydot(3) = y(4)
      ydot(4) = (1.0/r)*(ftheta + r*phidot**2*dsin(theta)*
     1 dcos(theta) - 2*rdot*thetadot)
      ydot(5) =y(6)
      ydot(6) = (1.0/(r*dsin(theta)))*(fphi - 2.0 * r * thetadot*
     1 phidot* dcos(theta) - 2*rdot*phidot*dsin(theta))

      return
      end

       subroutine forces(neq,t,y,fr,ftheta,fphi,velnorm,aero_forces,
     1 g_forces)
implicit real*8(a-h,o-z)
      dimension y(30)
      dimension  fgravity(3),faero(3), vel(3), total_force(3)
      dimension e1(3),e2(3), dirn_lift(3)
      r = y(1)
      rdot = y(2)
      theta = y(3)
      thetadot = y(4)
      phi = y(5)
      phidot = y(6)
      earth_radius  = y(7)
      satmass = y(8)
      alpha_init = y(9)
      clift = y(10)
      cdrag = y(11)
      cdrogue_drag = y(12)
```

```
      cmain_drag = y(13)
      varea = y(14)
      drogue_area = y(15)
      cmain_area = y(16)
      drogue_mach = y(17)
      cmain_altitude = y(18)
      density_sl = y(19)
      alpha_rate = y(20)
      wind_speed = y(21)
      wind_component = -1.0* wind_speed/(r*dsin(theta))

      pi = 4.0*atan(1.0)
      degree = pi/180.0
      earth_rot = (1.0/240.0)*degree
      phidot = phidot + earth_rot + wind_component



c compute forces due to gravity
fgravity(1) = -9.81*(earth_radius)**2/r**2
fgravity(2) = 0.0
fgravity(3) = 0.0

c compute forces due to aerodynamic effects
density=density_sl*dexp(-1.0*(r-earth_radius)/7162.8)

c compute reference area for drag
if (r .le. (earth_radius + cmain_altitude)) then
    ref_area = cmain_area + drogue_area + varea
            cdfinal = (cdrag*varea + cdrogue_drag*drogue_area
     1          + cmain_drag*cmain_area)/(varea + drogue_area +
     2           cmain_area)
      else
ref_area=varea
cdfinal=cdrag
endif
cdfinal = cdfinal*(-1.0)
c velocity vector
vel(1)=rdot
vel(2)=r*thetadot
vel(3)=r*phidot*dsin(theta)
velnorm = enorm(3,vel)

c multiplying factor for aerodynamic effects
faero_factor=density*velnorm**2 * ref_area/(2.0*satmass)
```

```
c vectors e1 and e2 for getting lift vector direction
e1(1) = - r*thetadot
e1(2) = rdot
e1(3) = 0.0
e1norm=enorm(3,e1)
e2(1) = - r*rdot*phidot*dsin(theta)
e2(2) = - r**2 * thetadot*phidot*dsin(theta)
e2(3) = rdot**2 + r**2*thetadot**2
e2norm = enorm(3,e2)
alphat = alpha_init + alpha_rate*degree*t
dirn_lift(1) = (e1(1)/e1norm)*dsin(alphat) +
     1 (e2(1)/e2norm)*dcos(alphat)
dirn_lift(2) = (e1(2)/e1norm)*dsin(alphat) +
     1 (e2(2)/e2norm)*dcos(alphat)
dirn_lift(3) = (e1(3)/e1norm)*dsin(alphat) +
     1 (e2(3)/e2norm)*dcos(alphat)

c compute aerodynamic forces
        faero(1) = faero_factor*(cdfinal*vel(1)/velnorm + clift*
     1  dirn_lift(1))
        faero(2) = faero_factor*(cdfinal*vel(2)/velnorm + clift*
     1  dirn_lift(2))
        faero(3) = faero_factor*(cdfinal*vel(3)/velnorm + clift*
     1  dirn_lift(3))
c magnitude of the aerodynamic forces
  aero_forces=enorm(3,faero)

c finally, the three force components..

c do 92 i=1,3
c faero(i) = 0.0
c 92 continue
fr     = fgravity(1) + faero(1)
ftheta = fgravity(2) + faero(2)
fphi   = fgravity(3) + faero(3)

total_force(1) = fr
total_force(2) = ftheta
total_force(3) = fphi
c magnitude of the acceleration
g_forces=enorm(3,total_force)/9.81

return
end
```

```
      DOUBLE PRECISION FUNCTION ENORM(N,X)
      INTEGER N
      DOUBLE PRECISION X(N)
C     **********
C
C     FUNCTION ENORM
C
C     GIVEN AN N-VECTOR X, THIS FUNCTION CALCULATES THE
C     EUCLIDEAN NORM OF X.
C
C     THE EUCLIDEAN NORM IS COMPUTED BY ACCUMULATING THE SUM OF
C     SQUARES IN THREE DIFFERENT SUMS. THE SUMS OF SQUARES FOR THE
C     SMALL AND LARGE COMPONENTS ARE SCALED SO THAT NO OVERFLOWS
C     OCCUR. NON-DESTRUCTIVE UNDERFLOWS ARE PERMITTED. UNDERFLOWS
C     AND OVERFLOWS DO NOT OCCUR IN THE COMPUTATION OF THE UNSCALED
C     SUM OF SQUARES FOR THE INTERMEDIATE COMPONENTS.
C     THE DEFINITIONS OF SMALL, INTERMEDIATE AND LARGE COMPONENTS
C     DEPEND ON TWO CONSTANTS, RDWARF AND RGIANT. THE MAIN
C     RESTRICTIONS ON THESE CONSTANTS ARE THAT RDWARF**2 NOT
C     UNDERFLOW AND RGIANT**2 NOT OVERFLOW. THE CONSTANTS
C     GIVEN HERE ARE SUITABLE FOR EVERY KNOWN COMPUTER.
C
C     THE FUNCTION STATEMENT IS
C
C       DOUBLE PRECISION FUNCTION ENORM(N,X)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C     SUBPROGRAMS CALLED
C
C       FORTRAN-SUPPLIED ... DABS,DSQRT
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I
      DOUBLE PRECISION AGIANT,FLOATN,ONE,RDWARF,RGIANT,S1,S2,S3,XABS,
     *                 X1MAX,X3MAX,ZERO
```

```
      DATA ONE,ZERO,RDWARF,RGIANT /1.0D0,0.0D0,3.834D-20,1.304D19/
      S1 = ZERO
      S2 = ZERO
      S3 = ZERO
      X1MAX = ZERO
      X3MAX = ZERO
      FLOATN = N
      AGIANT = RGIANT/FLOATN
      DO 90 I = 1, N
         XABS = DABS(X(I))
         IF (XABS .GT. RDWARF .AND. XABS .LT. AGIANT) GO TO 70
            IF (XABS .LE. RDWARF) GO TO 30
C
C              SUM FOR LARGE COMPONENTS.
C
               IF (XABS .LE. X1MAX) GO TO 10
                  S1 = ONE + S1*(X1MAX/XABS)**2
                  X1MAX = XABS
                  GO TO 20
   10          CONTINUE
                  S1 = S1 + (XABS/X1MAX)**2
   20          CONTINUE
               GO TO 60
   30       CONTINUE
C
C              SUM FOR SMALL COMPONENTS.
C
               IF (XABS .LE. X3MAX) GO TO 40
                  S3 = ONE + S3*(X3MAX/XABS)**2
                  X3MAX = XABS
                  GO TO 50
   40          CONTINUE
                  IF (XABS .NE. ZERO) S3 = S3 + (XABS/X3MAX)**2
   50          CONTINUE
   60       CONTINUE
            GO TO 80
   70    CONTINUE
C
C           SUM FOR INTERMEDIATE COMPONENTS.
C
            S2 = S2 + XABS**2
   80    CONTINUE
   90    CONTINUE
C
C     CALCULATION OF NORM.
C
```

```
      IF (S1 .EQ. ZERO) GO TO 100
         ENORM = X1MAX*DSQRT(S1+(S2/X1MAX)/X1MAX)
         GO TO 130
  100 CONTINUE
         IF (S2 .EQ. ZERO) GO TO 110
            IF (S2 .GE. X3MAX)
     *         ENORM = DSQRT(S2*(ONE+(X3MAX/S2)*(X3MAX*S3)))
            IF (S2 .LT. X3MAX)
     *         ENORM = DSQRT(X3MAX*((S2/X3MAX)+(X3MAX*S3)))
            GO TO 120
  110    CONTINUE
            ENORM = X3MAX*DSQRT(S3)
  120    CONTINUE
  130 CONTINUE
      RETURN
C
C     LAST CARD OF FUNCTION ENORM.
      END
```

Function D1MACH

The following machine specific constants were used for both NeXT Station Turbo and for HP 9000/710 workstations.

```
      DOUBLE PRECISION FUNCTION D1MACH(I)
C
C  DOUBLE-PRECISION MACHINE CONSTANTS
C
C  D1MACH( 1) = B**(EMIN-1), THE SMALLEST POSITIVE MAGNITUDE.
C
C  D1MACH( 2) = B**EMAX*(1 - B**(-T)), THE LARGEST MAGNITUDE.
C
C  D1MACH( 3) = B**(-T), THE SMALLEST RELATIVE SPACING.
C
C  D1MACH( 4) = B**(1-T), THE LARGEST RELATIVE SPACING.
C
C  D1MACH( 5) = LOG10(B)
C
C  TO ALTER THIS FUNCTION FOR A PARTICULAR ENVIRONMENT,
C  THE DESIRED SET OF DATA STATEMENTS SHOULD BE ACTIVATED BY
C  REMOVING THE C FROM COLUMN 1.
C  ON RARE MACHINES A STATIC STATEMENT MAY NEED TO BE ADDED.
C  (BUT PROBABLY MORE SYSTEMS PROHIBIT IT THAN REQUIRE IT.)
C
C  FOR IEEE-ARITHMETIC MACHINES (BINARY STANDARD), ONE OF THE FIRST
```

```
C  TWO SETS OF CONSTANTS BELOW SHOULD BE APPROPRIATE.  IF YOU DO NOT
C  KNOW WHICH SET TO USE, TRY BOTH AND SEE WHICH GIVES PLAUSIBLE
C  VALUES.
C
C  WHERE POSSIBLE, DECIMAL, OCTAL OR HEXADECIMAL CONSTANTS ARE USED
C  TO SPECIFY THE CONSTANTS EXACTLY.  SOMETIMES THIS REQUIRES USING
C  EQUIVALENT INTEGER ARRAYS.  IF YOUR COMPILER USES HALF-WORD
C  INTEGERS BY DEFAULT (SOMETIMES CALLED INTEGER*2), YOU MAY NEED TO
C  CHANGE INTEGER TO INTEGER*4 OR OTHERWISE INSTRUCT YOUR COMPILER
C  TO USE FULL-WORD INTEGERS IN THE NEXT 5 DECLARATIONS.
C
C  COMMENTS JUST BEFORE THE END STATEMENT (LINES STARTING WITH *)
C  GIVE C SOURCE FOR D1MACH.
C
      INTEGER SMALL(2)
      INTEGER LARGE(2)
      INTEGER RIGHT(2)
      INTEGER DIVER(2)
      INTEGER LOG10(2)
      INTEGER SC
C
      DOUBLE PRECISION DMACH(5)
C
      EQUIVALENCE (DMACH(1),SMALL(1))
      EQUIVALENCE (DMACH(2),LARGE(1))
      EQUIVALENCE (DMACH(3),RIGHT(1))
      EQUIVALENCE (DMACH(4),DIVER(1))
      EQUIVALENCE (DMACH(5),LOG10(1))
C
C     MACHINE CONSTANTS FOR BIG-ENDIAN IEEE ARITHMETIC (BINARY FORMAT)
C     MACHINES IN WHICH THE MOST SIGNIFICANT BYTE IS STORED FIRST,
C     SUCH AS THE AT&T 3B SERIES, MOTOROLA 68000 BASED MACHINES (E.G.
C     SUN 3), AND MACHINES THAT USE SPARC, HP, OR IBM RISC CHIPS.
C
       DATA SMALL(1),SMALL(2) /    1048576,          0 /
       DATA LARGE(1),LARGE(2) / 2146435071,         -1 /
       DATA RIGHT(1),RIGHT(2) / 1017118720,          0 /
       DATA DIVER(1),DIVER(2) / 1018167296,          0 /
       DATA LOG10(1),LOG10(2) / 1070810131, 1352628735 /, SC/987/
C
C
C  ***  ISSUE STOP 779 IF ALL DATA STATEMENTS ARE COMMENTED...
      IF (SC .NE. 987) STOP 779
C  ***  ISSUE STOP 778 IF ALL DATA STATEMENTS ARE OBVIOUSLY WRONG...
      IF (DMACH(4) .GE. 1.0D0) STOP 778
      IF (I .LT. 1  .OR.  I .GT. 5) GOTO 999
```

```
      D1MACH = DMACH(I)
      RETURN
  999 WRITE(*,1999) I
 1999 FORMAT(' D1MACH - I OUT OF BOUNDS',I10)
      STOP
      END
```