

1N-18
12356
P.191

NASA CONTRACTOR
REPORT

NASA CR-184153

MLIBLAST - A PROGRAM TO EMPIRICALLY PREDICT
HYPERVELOCITY IMPACT DAMAGE TO THE SPACE STATION

By William K. Rule
University of Alabama - Tuscaloosa

Final Report

May 1991

Prepared for
NASA-Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

(NASA-CR-184153) MLIBlast: A PROGRAM TO
EMPIRICALLY PREDICT HYPERVELOCITY IMPACT
DAMAGE TO THE SPACE STATION Final Report
(Alabama Univ.) 191 p

CSCL 22B

N91-22363

Unclas
0012356

G3/18

11

11

11

11

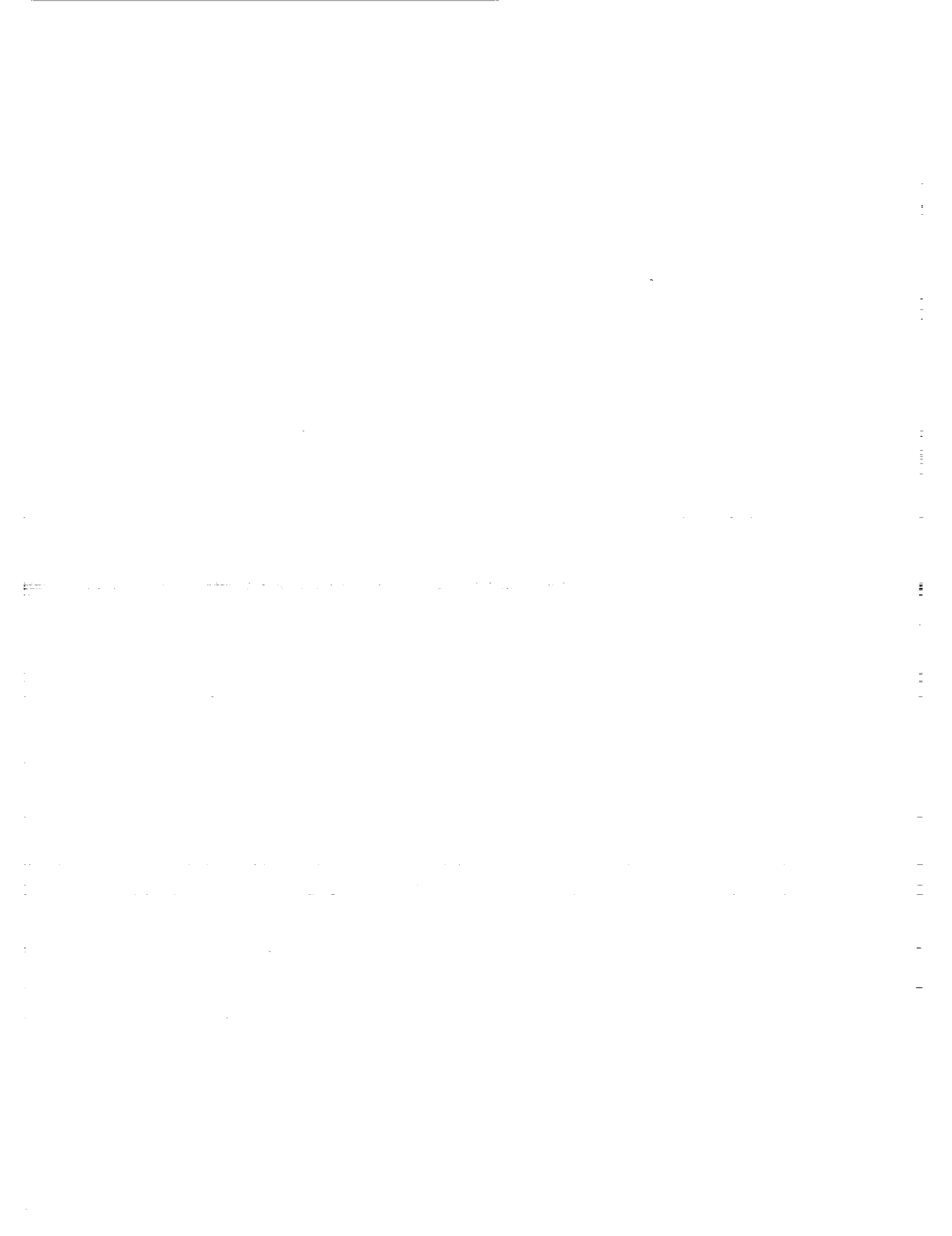
11

11



Report Documentation Page

1. Report No. NASA CR-184153	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle MLIBlast - A Program to Empirically Predict Hypervelocity Impact Damage to the Space Station		5. Report Date May 1991	
		6. Performing Organization Code	
7. Author(s) William K. Rule		8. Performing Organization Report No.	
		10. Work Unit No.	
9. Performing Organization Name and Address Department of Engineering Mechanics University of Alabama - Tuscaloosa Box 870278 Tuscaloosa, Alabama 35487-0278		11. Contract or Grant No. NAG8-123 (10)	
		13. Type of Report and Period Covered Final Contractor Report	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code	
		15. Supplementary Notes Prepared by Structures and Dynamics Laboratory, Science and Engineering Directorate, NASA, George C. Marshall Space Flight Center, Marshall Space Flight Center, Alabama	
16. Abstract <p>This report describes MLIBlast, which consists of a number of DOC PC based Microsoft BASIC program modules written to provide spacecraft designers with empirical predictions of space debris damage to orbiting spacecraft. The Spacecraft wall configuration is assumed to consist of multilayer insulation (MLI) placed between a Whipple style bumper and the pressure wall. Predictions are based on data sets of experimental results obtained from simulating debris impacts on spacecraft. One module of MLIBlast facilitates creation of the database of experimental results that is used by the damage prediction modules of the code. The user has a choice of three different prediction modules to predict damage to the bumper, the MLI, and the pressure wall. One prediction module is based on fitting polynomials to the experimental data. Another prediction module fits functions based on nondimensional parameters through the data. The last prediction technique is a unique approach developed for this project that is based on weighting the experimental data according to the distance from the design point.</p>			
17. Key Words (Suggested by Author(s)) Orbital Debris Empirical Analysis Hypervelocity Impact		18. Distribution Statement Unclassified-Unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 192	22. Price NTIS



ACKNOWLEDGMENTS

The author wishes to thank Pedro Rodriguez, Scott Hill, Paul Thompson, Sherman Avans and Jennifer Horn of Marshall Space Flight Center for providing the experimental data and background information, and for enthusiastically supporting this effort.



TABLE OF CONTENTS	PAGE
NOMENCLATURE	IV
1. INTRODUCTION	1
2. SOFTWARE USER GUIDE	4
3. THE INVERSE R PREDICTION TECHNIQUE	12
4. THE POLYNOMIAL FUNCTION PREDICTION TECHNIQUE	32
5. THE NONDIMENSIONAL PARAMETER PREDICTION TECHNIQUE	35
6. A COMPARISON OF THE ACCURACY OF THE PREDICTION TECHNIQUES	40
7. CONCLUSIONS AND RECOMMENDATIONS	47
REFERENCES	48
APPENDIX 1 A PREDICTION TECHNIQUE BASED ON AN EXTENSION OF THE ISOPARAMETRIC FORMULATION OF THE FINITE ELEMENT METHOD - A MASTER'S THESIS BY MR. P. WANG	51
APPENDIX 2 A LISTING OF THE COMPUTER CODE. (LISTING NOT PROVIDED IF FLOPPY DISKS INCLUDED)	127

NOMENCLATURE

- C_i = coefficients in polynomial or nondimensional prediction functions
- D_i = value of dependent variable (impact damage) at i^{th} data point
- D_{MAX} = maximum diameter of the bumper hole
- D_{MIN} = minimum diameter of the bumper hole
- D_{MLI} = average diameter of the hole in the MLI
- D_p = diameter of projectile
- D_{pw} = average diameter of the pressure wall hole.
- D_s = bumper stand-off distance
- D = estimated value of dependent variable (impact damage) at an interpolation or prediction point
- E = elastic modulus of the bumper plate material
- M = number of data points in database or number of material properties in each record of the materials data file
- N = number of independent variables (impact parameters)
- rn# = random number used to generate test function
- R^2 = coefficient of determination
- R_i = distance from i^{th} data point to interpolation or prediction point
- S = length of influence of a data point
- T_b = bumper thickness
- T_{pw} = pressure wall thickness
- V_s = speed of sound in the bumper material = $\sqrt{E/\rho}$
- $x_{j,i}$ = j^{th} coordinate (independent variable) of i^{th} data point
- $x_{j,\text{INT}}$ = j^{th} coordinate (independent variable) of interpolation or prediction point
- $\Delta x_{j,i} = x_{j,i} - x_{j,\text{INT}}$
- V = projectile velocity
- ϕ = impact angle
- ρ = mass density of the bumper plate material
- θ = weighting factor of a data point

1. INTRODUCTION

There are many engineering applications where predictions of the behavior of a physical system must be made based on a database of experimental results. In these instances, either the phenomenon is too complicated to treat analytically or numerically, or the funding, expertise or time required to do so is not available. Empirical approaches of this nature have always played a fundamental role in engineering design.

The purpose of this project was to develop a DOS microcomputer-based computer program to empirically predict hypervelocity impact damage to the Space Station from space debris. The main goal was to predict damage to the multilayer insulation (MLI). However, to extend the usefulness of the program, damage to other components of the Space Station wall are predicted as well. The program is intended to be an easy to use design tool for trade studies on debris protection strategies for the Space Station. The predictions are made based on a database of experimental results.

MSFC has a light gas gun that can launch 2.5-12.7 mm projectiles at speeds of 2-8 km/s.¹ Work is currently in progress at MSFC to qualify the orbital debris protection system under development by Boeing for Space Station Freedom. A schematic of the protection system is shown in Fig. 1.1. It is based on the classical sacrificial bumper approach first suggested by Whipple.² The purpose of the bumper is to break-up or ideally vaporize the projectile (space debris or micrometeoroids) so that the pressurized spacecraft behind the bumper is impacted with a series of fine particles rather than a single large particle.

The parameters associated with the impact data are illustrated in Fig. 1.1. The projectiles were initially spherical and typically constructed of 1100 aluminum. The bumper and the pressure walls were typically made from 6061-T6 and 2219-T87 aluminum, respectively. Some tests have been run with different materials. If a number of different materials were used for the bumper and the other components in the same database, then the number of independent degrees of freedom would be increased dramatically. This is because material properties, such as densities and melting points, would also have had to be accounted for.

There are five computer programs that were developed for this project. Details of how to use the program are provided in section 2. A main program called MLIBLAST serves as a shell to run the other four programs. A program called DATABASE is provided to assist the user in creating and adding to the database file of experimental results.

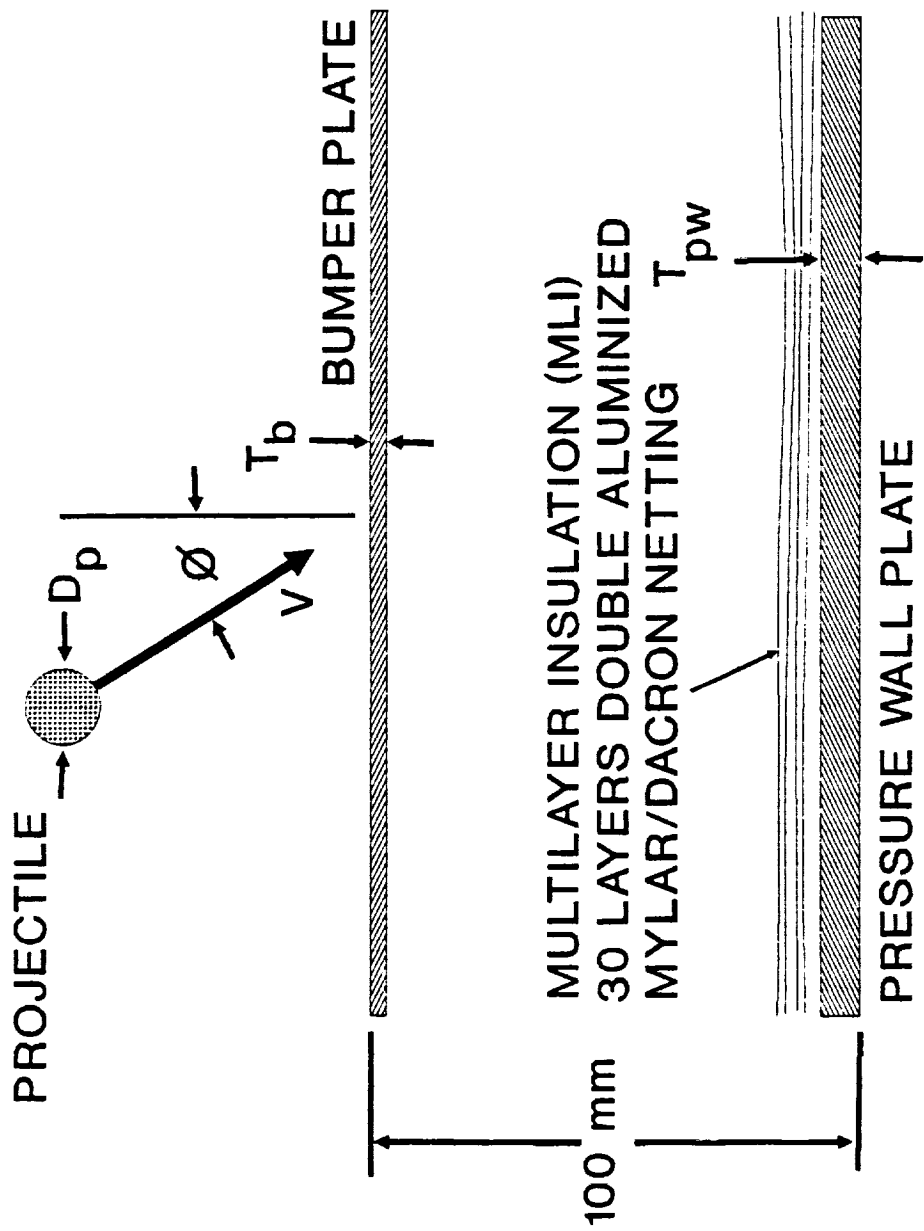


Fig. 1.1 Schematic drawing of impact specimen.

The remaining three programs provide predictions of impact damage to the bumper, the multilayer insulation (MLI), and the pressure wall plate. Program INVRMETH uses a unique prediction technique, called the inverse R method, that was developed for the purposes of this project. The theoretical basis of this method is described in section 3.

As described in section 4, program POLYMETH makes predictions by fitting simple polynomials through a subset of the data points. A more sophisticated form of polynomial prediction technique using the isoparametric formulation of the finite element method (FEM) as a basis was also attempted during the course of this project. This FEM based software was found to be somewhat unreliable for making predictions from the impact data that is currently available and so it was not included with this software. The interested reader can consult Appendix 1 for a discussion of this method.

The last prediction program, NONDIMEN makes predictions based on nondimensionalized functions that were developed by others and extended by the author for application here. These functions are described in section 5. The relative accuracies of these three prediction schemes are compared using an actual impact data set in section 6.

Lists of conclusions and recommendations derived from this research project are given in section 7.

A listing of the Microsoft BASIC source code is provided in Appendix 2 or is provided on the computer disks.

2. SOFTWARE USER GUIDE

The software developed for this project was written in Microsoft BASIC for DOS and compiled using the Microsoft BASIC Professional Development System compiler (version 7.0) for DOS based micro-computers. Approximately 0.5 MB of hard disk space, an EGA or VGA graphics card and monitor, and an Intel 80286, 80386 or 80486 CPU is required to run the software. A math coprocessor is desirable, but not required. The source code may be modified and recompiled using Microsoft QuickBASIC if desired.

The software is provided on two 5.25", 360K computer disks. An annotated listing of the contents of the computer disks follows:

DISK 1

DATABASE.BAS - source code for the data base program (ASCII).
DATABASE.EXE - compiled version of the database program.
INVRMETH.BAS - source code for the inverse R method damage prediction program (ASCII).
INVRMETH.EXE - compiled version of the inverse R method damage prediction program.
MATERIAL.DAT - a typical database file of material properties which is used by the INVRMETH program (ASCII).
MLI.DAT - a typical database file of experimental results (ASCII).
MLIBLAST.BAS - source code for the main program that runs the other programs (ASCII).
MLIBLAST.EXE - compiled version of the main program.

DISK 2

NONDIMEN.BAS - source code for the nondimensional functions damage prediction program (ASCII).
NONDIMEN.EXE - compiled version of the nondimensional functions damage prediction program (ASCII).
POLYMETH.BAS - source code for the polynomial functions damage prediction program (ASCII).
POLYMETH.EXE - compiled version of the polynomial functions damage prediction program.

The software is installed by first creating a sub directory on the hard disk and then copying all of the files from the two computer disks supplied into that subdirectory. If disk space is a problem then the source code files

(filename.BAS) need not be copied. The program is started by typing MLIPLAST and following the prompts. More details on the program prompts are given below, but first the database files MATERIAL.DAT and MLI.DAT will be discussed.

MATERIAL.DAT is an example of a typical materials data file. Any valid DOS name can be used for this file. Thus, the user may have several of this type of data file in a directory for different purposes. A file of this nature is required while running the inverse R program. The materials data file is an ASCII file that can be created and modified using any standard text editor. The format of the file is as follows:

```

material property 1 name string
material property 2 name string
.
.
.
material property M name string
{
material 1 name string
material property 1 for material 1
material property 2 for material 1
.
.
.
material property M for material 1
}
.
.
.
        ▶ LISTING OF NAMES OF MATERIAL
        ▶ PROPERTIES TO BE MODELED (MAXIMUM OF 10)
        (25 CHARACTERS MAX)
.
.
.
        ▶ TYPICAL DATA RECORD
.
.
.
        ▶ ANY NUMBER OF DATA RECORDS MAY BE USED

```

A typical material data file is provided on the computer disks and is called MATERIAL.DAT. This file is reproduced below:

```

Density (lb/in^3)
Elastic Mod. (lb/in^2)
Ultimate Strgth (lb/in^2)
Sp. Heat (BTU/(lb-deg R))
Melting Temp (deg R)
{
1100
9.780E-2
1.000E7
1.600E4
2.140E-1
1.680E3
}
{
2219-T87
1.030E-1
1.050E7
6.300E4
2.050E-1
1.680E3
}

```

```
)  
{  
6061-T6  
9.800E-2  
9.900E6  
4.200E4  
2.100E-1  
1.680E3  
}
```

MATERIAL.DAT is set up to model the material properties: density, elastic modulus, ultimate strength, specific heat, and melting temperature. Other physical properties can be used to a maximum of 10. The units do not have to be included in the material property name string. MATERIAL.DAT contains three records of material data for materials: 1100, 2219-T87, and 6061-T6. The material names are treated as string variables and thus can be any combination of numbers and letters. Any number of records of material data may be included. The order of the material properties must be the same in every record and be as the material property name strings are listed. For instance, referring to file MATERIAL.DAT, the specific heat of material 2219-T87 is 2.050E-1.

The purpose of the material properties database file is to provide an efficient, yet very flexible scheme for inputting material property data into the inverse R method computer program. The user can easily change the material properties to be modeled without disturbing the database file of experimental results. If the materials used for the projectile, bumper and pressure wall do not vary in the database, then the contents of the material properties database file will have no effect on the damage predicted by the inverse R method program. The polynomial function method program assumes that the material properties do not vary in the database. The nondimensional function method program assumes that the material properties of the projectile and pressure wall do not vary in the database and inputs material properties associated with the bumper directly.

The other database file required for running the programs of MLIBLAST is associated with the experimental data. This file can be created (and enlarged) by running the DATABASE program from inside MLIBLAST or it can be created "by hand" using any standard text editor since it is an ASCII file. This file can be given any valid DOS file name. Currently, up to 100 data records can be placed in this file. The format for this file is as follows:

```
{  
Test Number String  
Test Agency String (SOURCE OF DATA)  
Test Date String
```

Bumper Material String (SAME FORMAT AS IN MATERIAL DATABASE FILE)
Bumper Thickness
Bumper Stand-Off
Pressure Wall Material (SAME FORMAT AS IN MATERIAL DATABASE FILE)
Pressure Wall Thickness
Projectile Material (SAME FORMAT AS IN MATERIAL DATABASE FILE)
Projectile Diameter
Impact Angle
Projectile Velocity
Bumper Hole Maximum Diameter (Major Axis) Dimension
Bumper Hole Minimum Diameter (Minor Axis) Dimension
MLI Mean Hole Diameter
MLI Mass Loss
Pressure Wall Hole Maximum Diameter (Major Axis) Dimension
Pressure Wall Hole Minimum Diameter (Minor Axis) Dimension
 }

.
 . ▶ AS MANY AS 99 MORE DATA RECORDS
 .

MLI.DAT is an example of an experimental database file. This file is provided on the computer disks. It contains information on the specimens recently used for thermal testing in Sunspot Thermal Vacuum Chamber. To help understand the format information given above, the first record of MLI.DAT is presented below for comparison:

```

{
1012
MSFC
05/08/90
6061-T6
.08
4
2219-T87
.125
1100
.313
0
6.72
.729
.729
2.2
.938
.6
.15
}
  
```

An overview of the menu choices available to the user of MLIBLAST will now be discussed. The program is started by typing MLIBLAST. The user is then provided with three options:

1. Add data to, or create a new experimental results database file. Selecting

this option will cause program **DATABASE** to run.

2. Make a prediction. This option involves running one of the three prediction programs: **INVRMETH**, **NONDIMEN**, or **POLYMETH**.

3. Quit **MLIBLAST**

The steps associated with running each of these programs will now be considered.

PROGRAM DATABASE:

Step 1 - Enter the name of an experimental results database file. Any valid DOS name can be used. If this file already exists, then the new data records will be appended to the end of it. **MLI.DAT** is an example of an experimental results data file. This file was provided on the computer disks.

Step 2 - Enter the appropriate data at the prompts. Press **ENTER** after the data has been typed in. If you make a mistake, then press the **F10** function key and then the **ENTER** key to redo the data input.

Step 3 - You will be prompted as to whether to write your previously entered data record information to your database file. This provides another way of not saving an input data record with errors. You will also be prompted as to whether to enter another data record. A response of *n* will cause you to exit from the database program. Note - the database file created is an ASCII file which can be edited with a standard text editor. Additional data records can be added to the experimental database file using the text editor (instead of program **DATABASE**) if so desired.

PROGRAM INVRMETH (SEE SECTION 3 FOR MORE DETAILS ON PROGRAM **INVRMETH**)

Step 1 - Input the names of the experimental database file and the material database file. The program will then read these files and present a summary of their contents on two computer screens. These summary screens are intended to help the user determine if the contents of the database file are appropriate for the desired prediction.

Step 2 - Select the quantity for which a prediction is to be made. This program is designed to make predictions for: bumper hole maximum and minimum hole dimensions, MLI average hole diameter, MLI mass loss, and pressure wall maximum and minimum hole dimensions.

Step 3 - Input the impact parameters (such as projectile diameter) associated with the desired prediction. Default values are provided in square brackets

for all inputs here except for impact angle. A default value is selected by simply pressing ENTER. The magnitude of the input impact parameter relative to the database average is indicated in round brackets. For instance if the projectile velocity for the prediction is twice that of the average projectile velocity in the experimental database file then the number 2 would appear in round brackets. Ideally, prediction parameter values should be close to the database average if reliable predictions are to be made. The round bracket numbers are intended to help the user assess the reliability of the prediction.

Step 4 - Review the results of the prediction. Here, the value for the prediction is given and the location of the prediction point along the prediction vector (see section 3) is indicated. Information on the polynomial fit through the 10 interpolation points (see section 3) is also provided. The user can also review the results of the prediction graphically. Here, the variation of the function to be predicted along the prediction vector is illustrated to assist the user in assessing the reliability of the prediction. If the function being predicted varies in an erratic fashion along the prediction vector then the prediction may be unreliable.

PROGRAM POLYMETH (SEE SECTION 4 FOR MORE DETAILS ON PROGRAM POLYMETH)

Step 1 - A warning screen is displayed indicating that the program only models the parameters: bumper thickness, projectile diameter, projectile velocity, and impact angle.

Step 2 - Input experimental results database filename. No material data file name is requested since it is assumed that material types will not vary in the database. The program will then read in the contents of the experimental results database file and display a summary of this data on the screen so that the user may assess its suitability with respect to the required predictions.

Step 3 - Select the desired prediction such as MLI hole diameter.

Step 4 - Input the impact parameters (such as bumper thickness) associated with the prediction.

Step 5 - The program now attempts to fit a complete linear polynomial through random subsets of the data as described in section 4. If the experimental data does not "span" the impact parameter space very well then the program may take a long time or may not be able to find a solution and make a prediction. In this case the user may press function key F1 to quit or may try to fit incomplete polynomial functions. To attempt to fit a "simpler"

function (only terms to second order retained) the user should press function key F2. Pressing F3 will cause the program to fit the "simplest" function which retains only terms to first order.

Step 6 - Analyze the results. The program will attempt to find five fits of the polynomial to random subsets of the experimental data. These will be displayed on the screen along with the average nondimensionalized distance (see section 4) of the data points used for the function fit. Intuitively, the prediction with the smallest average distance (data points closest to where the prediction is required) should be the most reliable. However, this may not be so if one of those data points happened to contain a large experimental error. Accordingly, a weighted average of the three predictions with the lowest mean distances is also presented. The weighting is based on the mean distances as described in section 4.

PROGRAM NONDIMEN (SEE SECTION 5 FOR MORE DETAILS ON PROGRAM NONDIMEN)

Step 1 - Input experimental database file name. No material database file name is required because it is assumed that material types will not vary in the database.

Step 2 - Input bumper elastic modulus (equal to 70E3 MPa for aluminum) and input the mass density of the bumper (equal to 2710 kg/m³ for aluminum). After these values have been input, information summarizing the contents of the experimental database file will be shown on the screen.

Step 3 - Parameters for the nonlinear function coefficient optimizer are input. The purpose of these parameters and recommended magnitudes are displayed on the computer screen.

Step 4 - At this time an iterative procedure is invoked to adjust the prediction function coefficients such that the coefficient of determination (R^2) is maximized. During this process the R^2 values are printed on the screen so that the user can assess the suitability of the functional form of prediction equations for fitting the experimental data.

Step 5 - On completion of the optimization process, the function coefficients and R^2 values are displayed on the screen to further assist the user in assessing the goodness of fit between the functions and the experimental data.

Step 5 - Input prediction parameters (such as bumper thickness) and make predictions.

In the next three sections more details on the prediction schemes are

presented. In section 6, the three prediction schemes are compared using the experimental data set associated with the impact specimens that were tested in the Sunspot Thermal Vacuum Chamber.

3. THE INVERSE R PREDICTION TECHNIQUE

The usual procedure for making predictions from experimental data is to assume some form for the equation relating the independent variables to the dependent variable. A function of this nature is described in section 5 of this report. The equation typically contains empirical coefficients, the values of which are determined from a fit to the experimental data.³⁻⁸ The method of least squares (maximizing the coefficient of determination, R^2) is an example of a popular technique for obtaining the coefficients from the experimental data. The final result is a closed form equation for making predictions.

This approach has been found to work very well for many engineering applications, however there are some disadvantages. A suitable form for the prediction equation must be developed. This is often difficult. Incorporating additional independent variables in an existing equation can pose problems. Usually, a well defined procedure for taking into account new experimental data is not put in place. Generally, a single set of empirical coefficients are used to make predictions over a fairly wide range of values of the independent variables. Thus, the best data in a database for making a prediction with a particular set of independent variables may not be used to best advantage. Also, it is usually difficult to assess the accuracy of the prediction.

In this section, a new method (called inverse R method) for making empirical predictions based on experimental data is discussed. The method uses a very general form of prediction equation that can be applied in the same manner to all problems. Thus, the user is not required to develop a suitable form for the prediction equation and additional independent variables can be easily incorporated. The new method is designed to work off a database that can be continuously updated as new experimental data becomes available. The method automatically takes advantage of the most appropriate data in the database for a given set of independent variables. The method provides diagnostics for assessing the accuracy of the prediction.

The new technique consists of four main steps which will now be described.

Step 1. Normalize the Independent Variables.

In general, the independent variables will vary greatly in magnitude. In hypervelocity impact work, dimensions can be of order 10 and velocities of order 10^6 . The new technique requires that all variables be of the same order of magnitude. This was accomplished by scaling the independent variables such

that their mean value is equal to unity. Other scaling methods could perhaps be used to improve the accuracy of this technique. For instance, the variables could be scaled such that predicted values of points in the database more closely match the measured values. This scaling technique was not tested. The dependent variables need not be scaled.

This technique works off a database that can and should be kept updated with the latest experimental data. Thus, the scaling factors will change as time progresses and the size of the database increases.

Step 2. Select a Series of Points in the Data Domain For Interpolation.

Two general requirements for prediction schemes are: the method should be capable of smoothing the data to (hopefully) cancel out the random scatter typically present in experimental measurements and the technique should allow for making reliable predictions outside of the domain of the measured data. Here, these requirements are satisfied by using the data to make ten interpolations from within the domain of the data, which are then used for predicting the dependent variable at some point of interest. The ten "interpolation" points should provide for sufficient smoothing of the data and also capture the trend characteristics of the data for extrapolation purposes, if an extrapolation is required. The number of interpolation points to use was selected on the basis of trial and error. Note, in some cases extrapolation can produce misleading results regardless of the extrapolation technique used.

Fig. 3.1 provides an illustration of how the interpolation points are selected for a hypothetical case with two independent variables. An identical approach is used for the case of an arbitrary number of independent variables. In Fig. 3.1, the independent variables are in the plane of the page and the dependent variable takes the form of a surface out of the plane of the page.

First, a "prediction" vector is drawn from the origin through the point in the domain where a prediction of the dependent variable is required, which is called the "target" point. Then the "min" and "max" points (Fig. 3.1) are located on the prediction vector by considering the intersection points of perpendiculars from the data points to the prediction vector. The closest intersection point to the origin defines the min point, and that of the farthest, the max point. Ten equally spaced points (interpolation points) on the prediction vector between the min and max point are then used for the next step in the prediction process. If the target point lies between the min and max points then an interpolation is required, otherwise an extrapolation

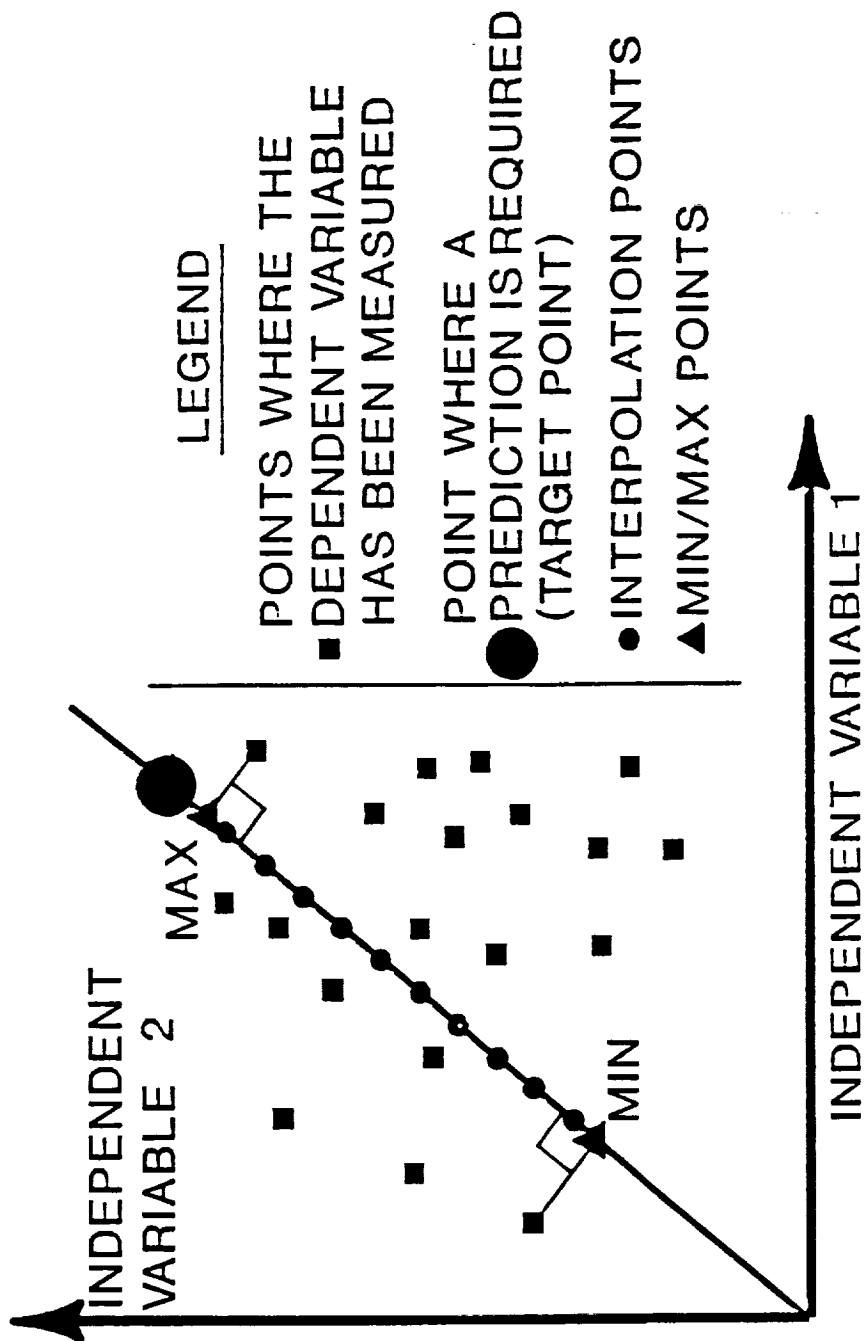


Fig. 3.1 Technique for selecting interpolation point locations for the case of two independent variables.

is required.

Step 3. Estimate Values of the Dependent Variable at Interpolation Points.

Next, values for the dependent variable must be estimated at the ten interpolation points. This is done as indicated in the following equation:

$$D = \frac{\sum_{i=1}^M \frac{D_i}{R_i^{N-1}}}{\sum_{i=1}^M \frac{1}{R_i^{N-1}}} \quad (3.1)$$

The R_i are determined by the usual formula for determining the "distance" between two points in a multidimensional space:

$$R_i^2 = \sum_{j=1}^N (x_{j,i} - x_{j,INT})^2 \quad (3.2)$$

The need for scaling the independent variables is evident from considering the form of Eq. (3.2).

The form of Eq. (3.1) will now be considered. It is assumed that if all measured data points are the same "distance" R from an interpolation point then all the measured data should be given equal weight. This situation is illustrated for the case of two independent variables ($N = 2$) in Fig. 3.2. This can be interpreted as saying that each data point has some "characteristic length of influence", S , that subtends an angle $\theta = S/R = S/R^{N-1}$ as indicated in Fig. 3.2. The θ can be taken as the weighting factor. For the constant R case shown in Fig. 3.2, all data points would be given the same weight. Fig. 3.3 illustrates the case for which the data points are considered to be equally valid (same S), but are located different "distances" from the interpolation point. Here, the weighting factors will be of the form $\theta_i = S/R_i^{N-1}$, and thus data points closer to the interpolation point will be given a higher weight. The value of the dependent variable at the interpolation point can be estimated from $D = \Sigma \theta_i D_i / \Sigma \theta_i$ which leads to Eq. (3.1) and hence this technique is given the name Inverse R Method. Note that a value for S is not required as it cancels out of the equation.

The three dimensional (three independent variables) application of this procedure leads to equations identical in form to those used for determining

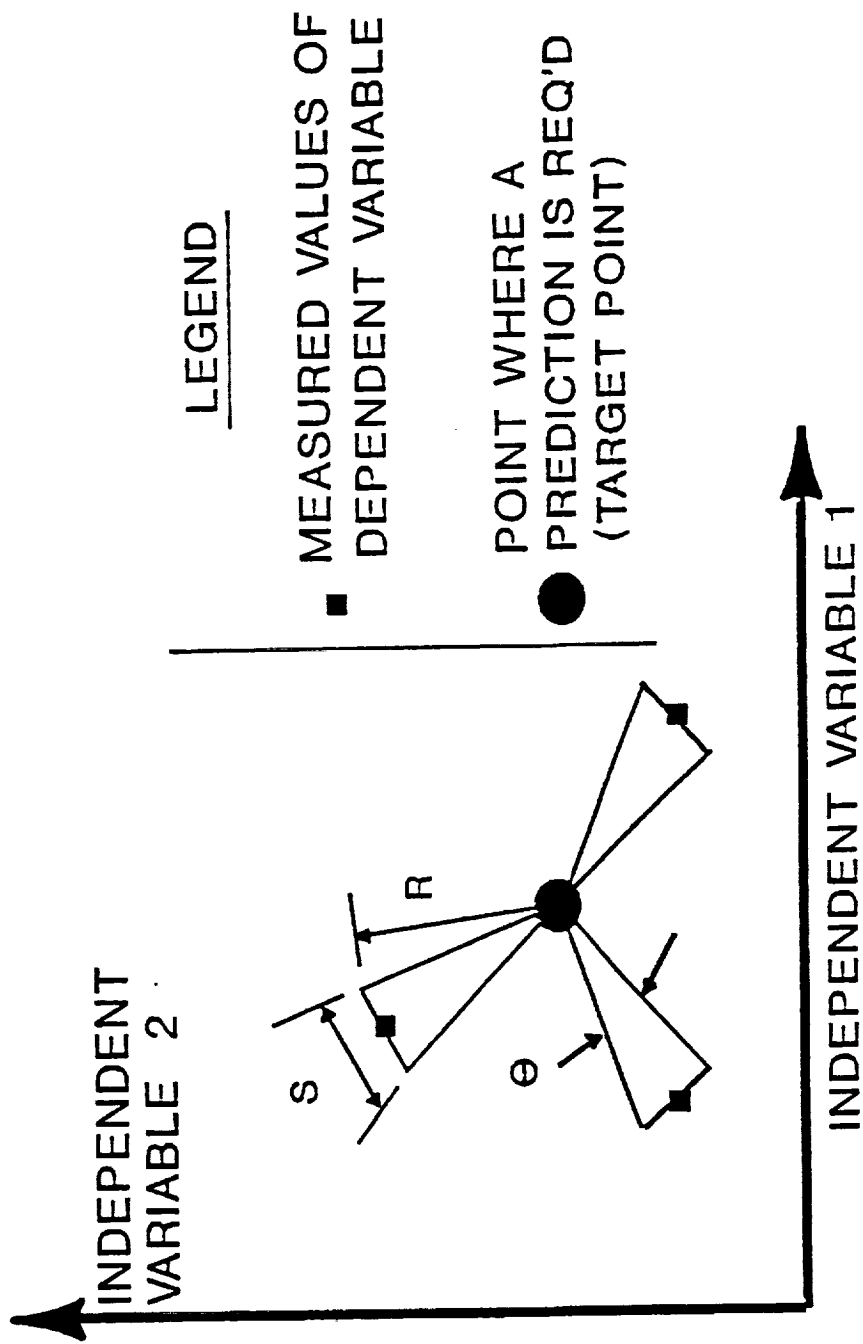


Fig. 3.2 Interpolation scheme for equally spaced data points.

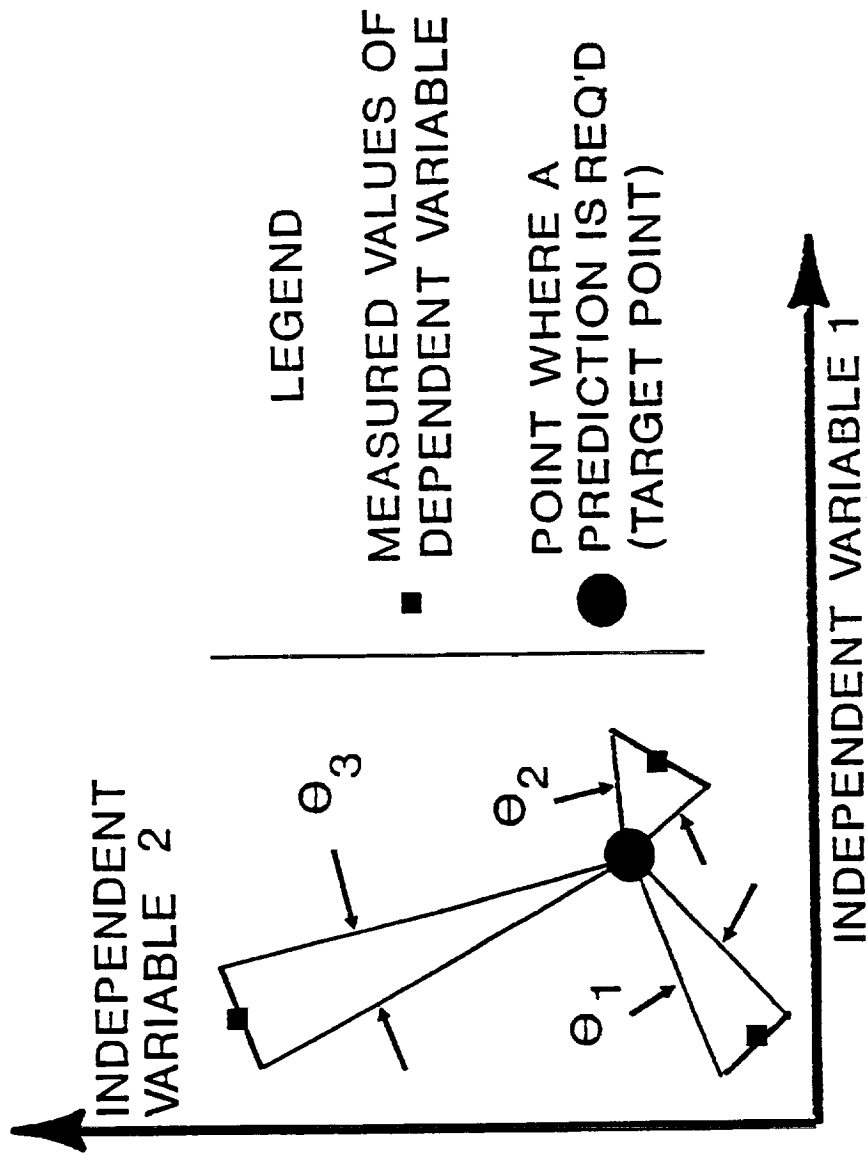


Fig. 3.3 Interpolation scheme for unequally spaced data points.

view factors in the field of radiation heat transfer.⁹ The method described herein can be interpreted as though the measured data points are "radiating" information to the interpolation point. The farther the data point is away, the weaker the "radiation" (lower weight given to the information). In principle, the method can easily be extended to any number of independent variables, N.

Step 4. Fit a Polynomial Through the Interpolation Points and Make Prediction.

The final step in the process involves fitting a polynomial through the ten interpolation points and then using the polynomial to make a prediction of the dependent variable at the target point. The polynomial describes how the dependent variable behaves as a function of distance along the prediction vector. By trial and error it was found that a fourth-order polynomial worked well. The polynomial could be used for interpolation or extrapolation depending on the location of the target point. There would of course be considerably more uncertainty in the prediction for the case of extrapolation. Errors in the ten interpolation points tend to get smoothed by the polynomial.

Reliability Diagnostics of the New Interpolation/Extrapolation Technique

The method proposed herein provides diagnostics to help assess the accuracy of the prediction. The computer program can provide the user with averages of the independent variables of the data currently in the database. If the independent variables associated with the target point are close to the database averages then the user can expect a more reliable result to be produced. The coefficient of determination of the polynomial fit through the ten interpolation points can be presented to the user to assess the scatter in the data. Finally, as shown schematically in Fig. 3.4, the ten interpolation points, the polynomial curve, and the prediction can be graphically illustrated on the computer screen to show how the dependent variable behaves as a function of distance along the prediction vector and also indicate the location of the target point along the prediction vector relative to the min and max points. If the dependent variable oscillates wildly, then unreliable predictions can be expected, particularly in the case of extrapolation. In most cases, target points located approximately half way between the min and max points produce the best results.

Testing the New Interpolation/Extrapolation Technique

The proposed method was tested by making predictions based on a database created using a known function so that the reliability of the prediction

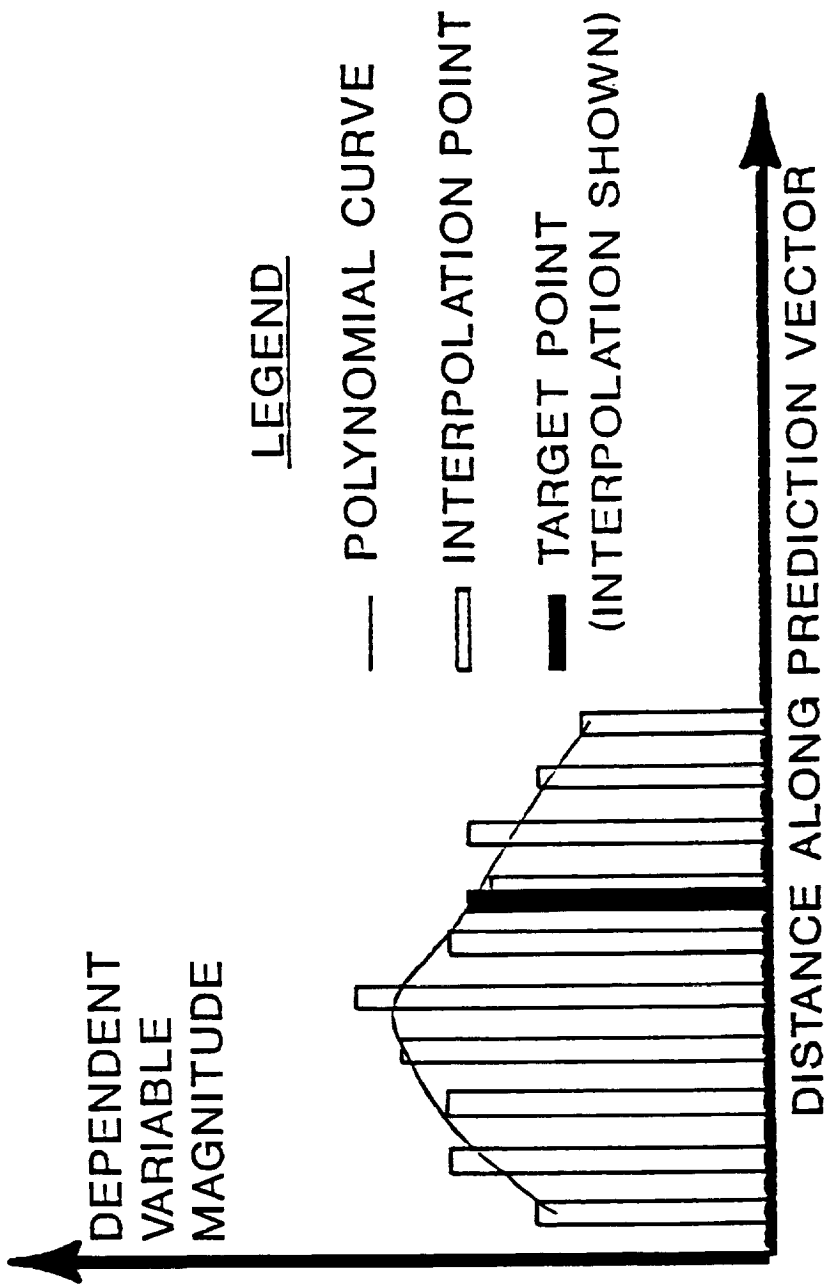


Fig. 3.4 Schematic of prediction reliability diagnostics.

could be assessed. The form of the function used was:

$$D_i = \prod_{j=1}^5 (rn\# + rn\# * x_{j,i} + rn\# * x_{j,i}^2) \quad (3.3)$$

A set of 15 random numbers, $rn\#$, was required - three for each of the 5 independent variables used to generate a data set. The random numbers were held constant during the data set creation so that a consistent set of dependent variables were generated. The intent here was to develop an unbiased, sophisticated and consistent set of data to provide an objective test of the interpolation/extrapolation technique.

The values of the independent variables used were obtained from a hypervelocity impact data set, Table 3.1. This data set was selected for two reasons. First, it seemed desirable to use a set of actual engineering data to provide a realistic test of the technique. Secondly, as is discussed below, the technique proposed herein did a poor job of predicting the behavior of some of the dependent variables of Table 3.1. Accordingly, it was of interest to determine if the nature of the data or the prediction technique was at fault for the poor predictions.

Eq. (3.3) was used with two sets of random numbers to generate two sets of consistent data. An analysis was done with each set of data as follows. Each record (data point) was temporarily removed from the database, a prediction made for the independent variables associated with that record, and then the record was returned to the database. This was done for all of the 35 records in the database. Thus, all predictions were made with the actual data point of interest removed from the database.

A typical set of results is shown in Fig. 3.5, where actual data values are plotted against their corresponding predictions. The dashed line in the figure is a linear least squares fit through the data of the form $y=mx+b$, where y is the prediction, x is the actual function value, and m and b are parameters to be fit. The coefficient of determination of this fit was 0.937. Assuming a functional form of $y=x$ produced a coefficient of determination of 0.934. Similarly, the other consistent data set produced coefficients of determination of 0.961 and 0.959, respectively. Ideally, the prediction, y , should exactly equal the actual value, x , which would result in a coefficient of determination of unity for the line $y=x$.

These results seem to be quite good considering that the dependent variables were reasonably complicated functions of fifteen random

Table 3.1 Experimental Data from Hypervelocity Impact Tests

Test ID	Bumper Thick. Tb(mm)	Pr. Wall Thick. Tpw(mm)	Proj. Diam. Dp(mm)	Impact Angle θ (deg)	Proj. Vel. V(km/s)	Bump. Maj.Ax. (mm)	Bump. Min.Ax. (mm)	MLI Pen. (cm ²)	MLI Per/Chr (cm ²)	Pr. Wall Maj.Ax. (mm)	Pr. Wall Min.Ax. (mm)
227A	0.81	1.60	6.35	45.00	5.52	15.24	11.68	32.26	51.61	13.46	9.65
227B	0.81	1.60	6.35	45.00	7.12	15.24	10.92	64.52	425.81	25.40	12.70
333	1.02	3.18	4.75	45.00	2.88	10.16	7.62	12.90	32.26	0.00	0.00
334	1.02	3.18	4.75	45.00	3.61	10.16	7.87	8.39	63.23	0.00	0.00
221C	1.02	3.18	4.75	45.00	4.57	11.43	9.14	19.35	51.61	0.00	0.00
221B	1.02	3.18	4.75	45.00	5.89	13.72	10.67	12.90	141.94	0.00	0.00
221A	1.02	3.18	4.75	45.00	6.36	12.19	10.16	12.90	148.39	0.00	0.00
336	1.02	3.18	6.35	45.00	4.47	13.46	10.67	64.52	129.03	14.99	6.35
201B	1.02	3.18	6.35	45.00	5.51	13.46	10.92	64.52	135.48	13.21	11.68
201C	1.02	3.18	6.35	45.00	7.21	13.46	6.35	16.13	161.29	2.54	2.54
203B	1.02	3.18	7.62	65.00	3.67	22.10	11.94	22.58	290.32	0.00	0.00
203A	1.02	3.18	7.62	65.00	6.45	23.88	13.46	29.03	232.26	0.00	0.00
003A	1.02	3.18	7.95	45.00	6.51	19.30	13.72	32.26	129.03	76.20	38.10
338	1.02	3.18	7.95	45.00	6.98	21.34	14.48	83.87	58.06	25.40	17.78
337	1.02	3.18	7.95	45.00	7.00	19.56	13.21	64.52	90.32	27.94	12.70
203F	1.02	3.18	8.89	65.00	3.04	24.89	12.45	13.55	270.97	0.00	0.00
339	1.02	3.18	9.53	45.00	6.49	21.08	17.53	129.03	258.06	50.80	38.10
218B	1.02	4.78	8.89	45.00	6.40	20.32	15.24	51.61	483.87	18.29	15.49
218C	1.02	4.78	8.89	45.00	6.76	21.34	14.99	70.97	270.97	30.73	10.16
230B	1.60	3.18	4.75	45.00	3.23	11.94	9.14	3.87	6.45	0.00	0.00
230A	1.60	3.18	4.75	45.00	4.41	12.19	9.91	6.45	48.39	0.00	0.00
301	1.60	3.18	6.35	45.00	2.95	13.72	10.92	4.26	118.32	0.00	0.00
205A	1.60	3.18	6.35	45.00	4.11	15.49	12.19	11.61	116.13	5.08	5.08
205B	1.60	3.18	6.35	45.00	4.59	16.51	12.45	24.52	335.48	5.08	5.08
205C	1.60	3.18	6.35	45.00	5.30	15.24	12.70	16.13	83.87	7.62	7.62
209B	1.60	3.18	6.35	65.00	6.40	22.10	13.21	5.16	193.55	0.00	0.00
209D	1.60	3.18	6.35	65.00	7.40	19.56	14.48	19.35	206.45	0.00	0.00
207A	1.60	3.18	7.62	65.00	5.86	22.35	14.99	11.61	329.03	4.06	4.06
207C	1.60	3.18	7.62	65.00	7.08	25.91	16.26	103.23	174.19	0.00	0.00
002B	1.60	3.18	7.95	45.00	6.39	20.57	15.75	129.03	77.42	5.59	5.59
211B	1.60	3.18	8.89	45.00	5.85	21.84	17.27	77.42	122.58	27.94	12.70
210B	1.60	3.18	8.89	65.00	5.70	28.70	16.76	41.94	96.77	3.18	3.18
210D	1.60	3.18	8.89	65.00	6.80	35.56	17.27	45.16	212.90	5.84	5.84
303B	1.60	4.06	7.95	45.00	4.34	18.03	14.48	32.90	362.26	0.00	0.00
303	1.60	4.06	7.95	45.00	4.59	18.54	14.73	17.03	166.84	0.00	0.00

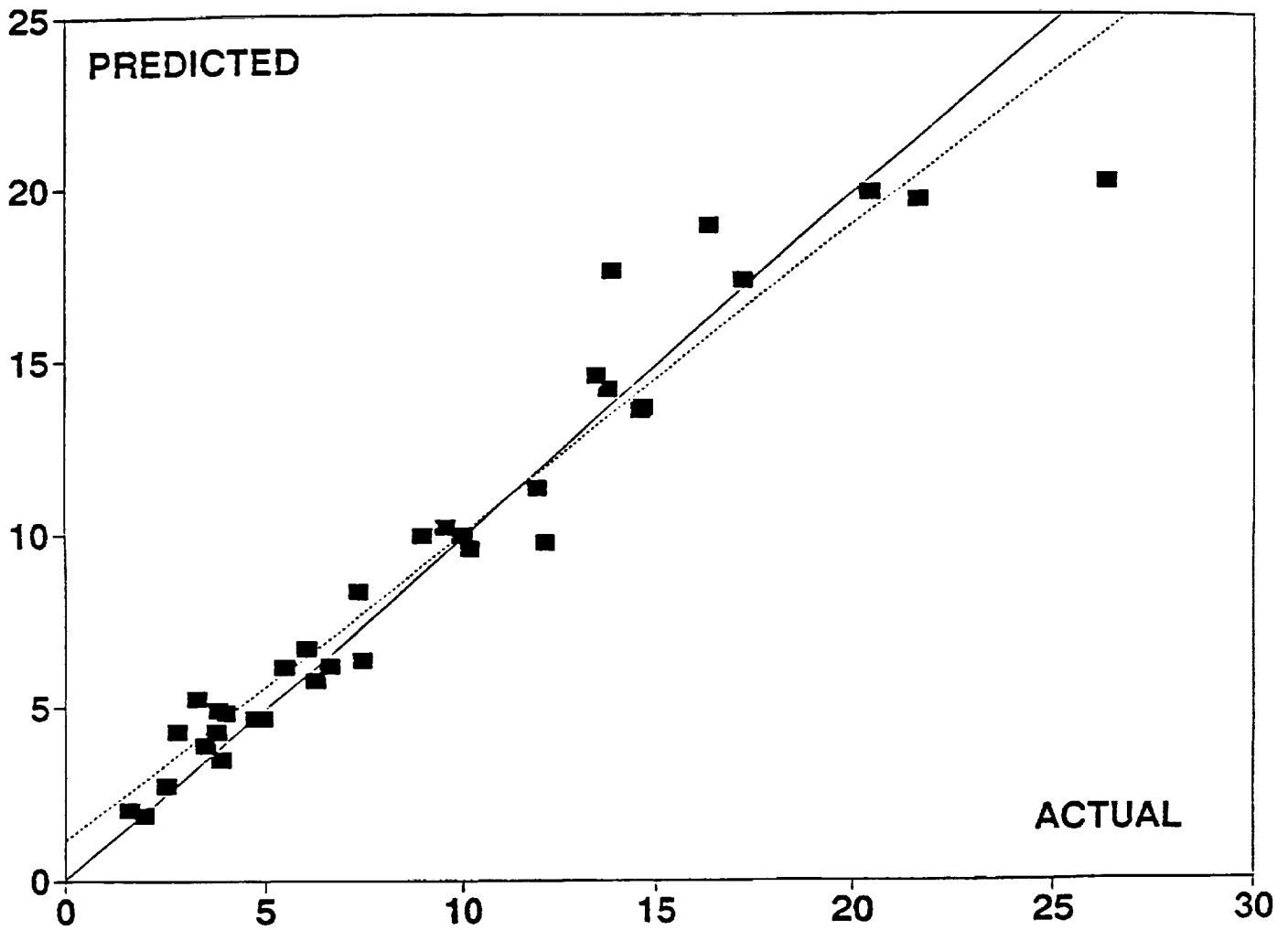


Fig. 3.5 Plot of actual VS predicted values using consistent data.

coefficients with five independent variables.

Applying the New Technique to Hypervelocity Impact Data

Personnel from the Structures and Dynamics Lab of Marshall Space Flight Center (MSFC) provided the author with a set of experimentally obtained hypervelocity impact data, Table 3.1. These impact tests were made with the multilayer insulation (MLI) placed directly against the pressure wall. The bumper plate was placed approximately 100 mm in front of the pressure wall plate. For this series of data, the pressure wall was unstressed. As listed in Table 3.1 and illustrated in Figure 1.1, the dependent variables measured included the major and minor axis dimensions of the bumper hole, the area of the hole clean through the MLI called the penetration area, the area of MLI outside of the penetration area obviously damaged by the impact called the perforated/charred area, and the major and minor axis dimensions of the pressure wall hole. Some comments will now be made on the characteristics of these dependent variables.

The bumper plate hole typically takes the form of a single, well defined, relatively smooth, elliptical hole. The greater the impact angle, the more elliptical the hole. It is not surprising that the bumper plate hole data is the most consistent of all the data given the relatively simple nature of the damage.

The remainder of the dependent variables are much more affected by characteristics of the fragmentation/vaporization process of the projectile than the bumper hole is. Launch loads typically cause the soft aluminum projectile to deform into a variety of nonspherical shapes. This effect, and the inevitable presence of a random assortment of microscopic flaws in the projectile and bumper, can cause large variations in the nature of the particles (from both the projectile and the bumper) that leave the back face of the bumper after the bumper-projectile impact. Thus, similar testing conditions can produce significantly different damage to the the MLI and the pressure wall.

There is a great deal of inconsistency in the MLI data. In addition to the random processes discussed previously, the inconsistency could be partly due to the difficulty in visually measuring the areas of damage (penetration and perforated/charred) because of the rough, irregular shapes of these areas.

Damage to the pressure wall typically consists of a large number of craters of various sizes, and possibly some penetrations. The craters and penetrations are typically distributed over a relatively large area as can be

seen in the photographs of Ref. 3. The data in Table 3.1 gives the dimensions of the largest penetration in the pressure wall, which would essentially depend on the the largest fragment that results from the bumper-projectile impact. As has been discussed, the same test conditions could produce a large variation in the size of the largest fragment and hence the size of the penetration. This leads to scatter in the pressure wall data.

The procedure described previously that was used to test the interpolation/extrapolation technique with the consistent data was also used with the experimental data of Table 3.1. Each record (data point) was temporarily removed from the database, a prediction made for the independent variables associated with that data point, and then the data point was returned to the database. The predicted versus actual data are shown in Figs. 3.6-3.11. Also drawn on these figures are solid lines indicating the ideal case of "predicted"="measured" The coefficients of determination associated with these predictions are given in Table 3.2. As can be seen from Table 3.2, the predictions for the bumper plate are acceptable. The predictions for the pressure wall are marginal, although Figs. 3.10 and 3.11 are somewhat pessimistic looking since a large number of good predictions were made for data located near the origin of the plots (no penetration case). The predictions of MLI damage are poor for penetration area and terrible for perforated/charred area.

Since the proposed interpolation/extrapolation technique produced acceptable results for both the consistent test functions of Eq. (3.3) and the bumper plate data of Table 3.1, the poor predictions of MLI and pressure wall damage are probably due to the scatter in the data produced by such effects as the distortion of the projectile during launch and the apparently random assortment of microscopic flaws in the projectile and the bumper. The proposed interpolation/extrapolation technique appears to be a useful tool for engineering design work.

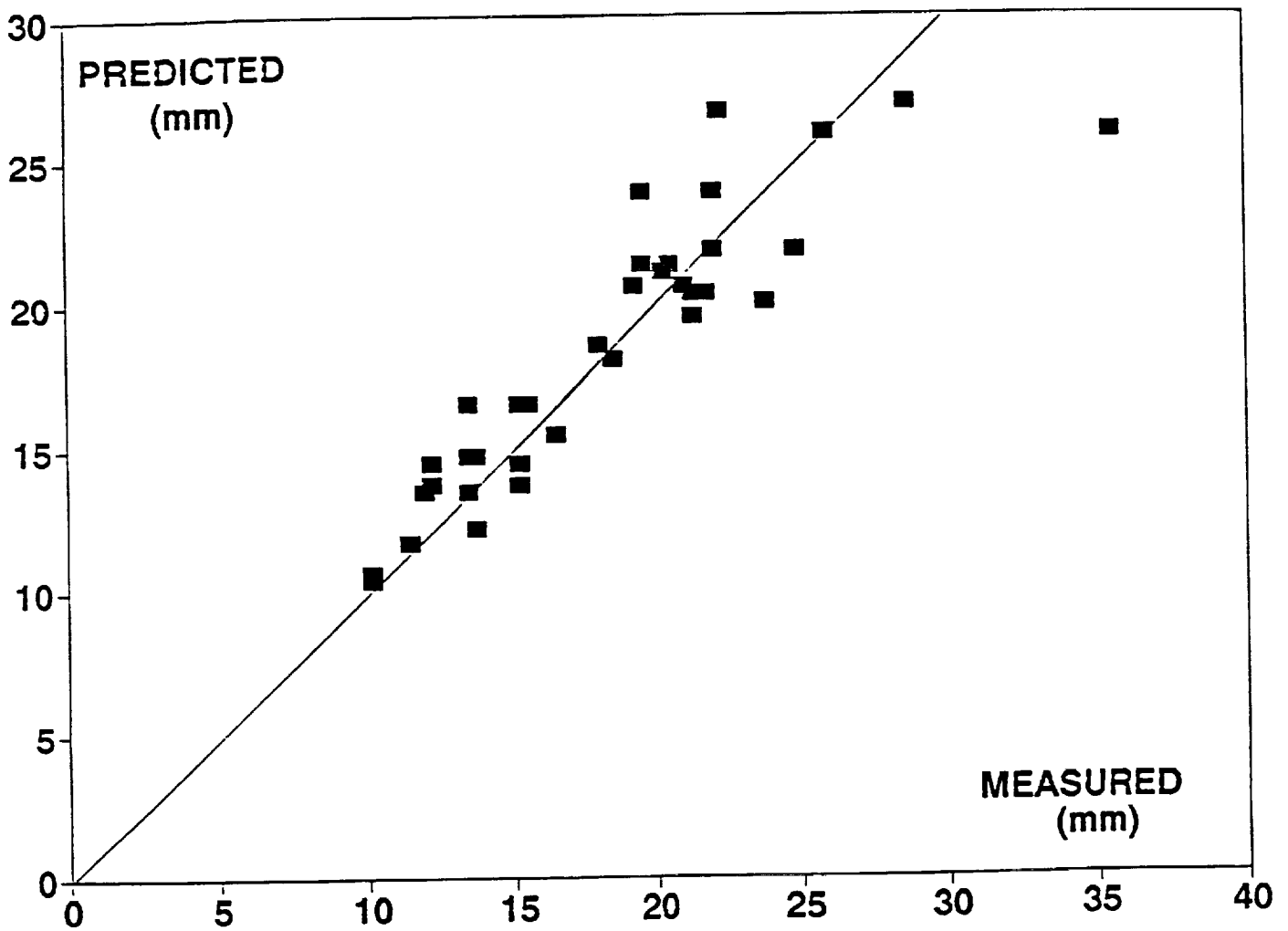


Fig. 3.6 Plot of measured VS predicted data for the major axis of the bumper plate hole.

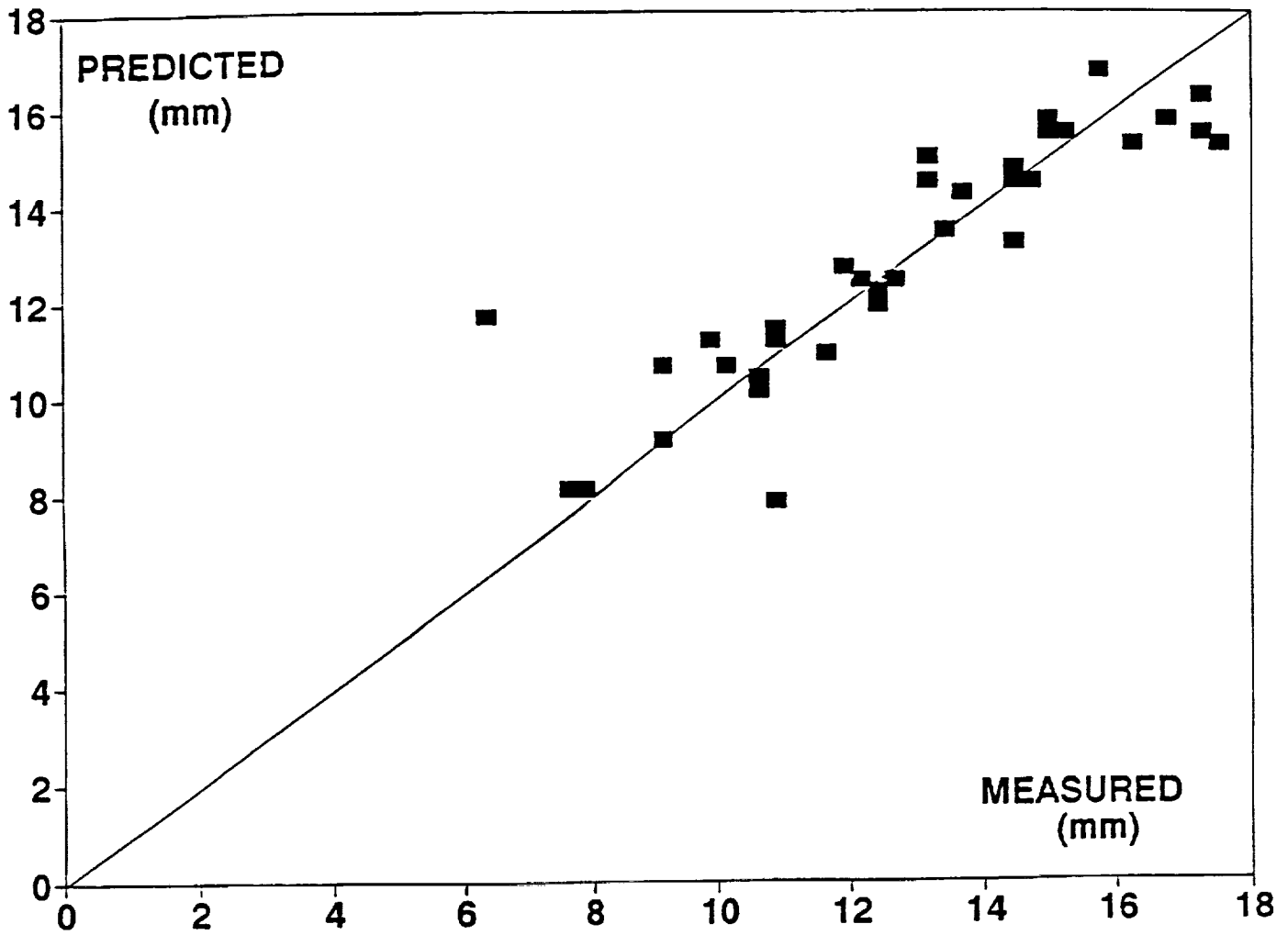


Fig. 3.7 Plot of measured VS predicted data for the minor axis of the bumper plate hole.

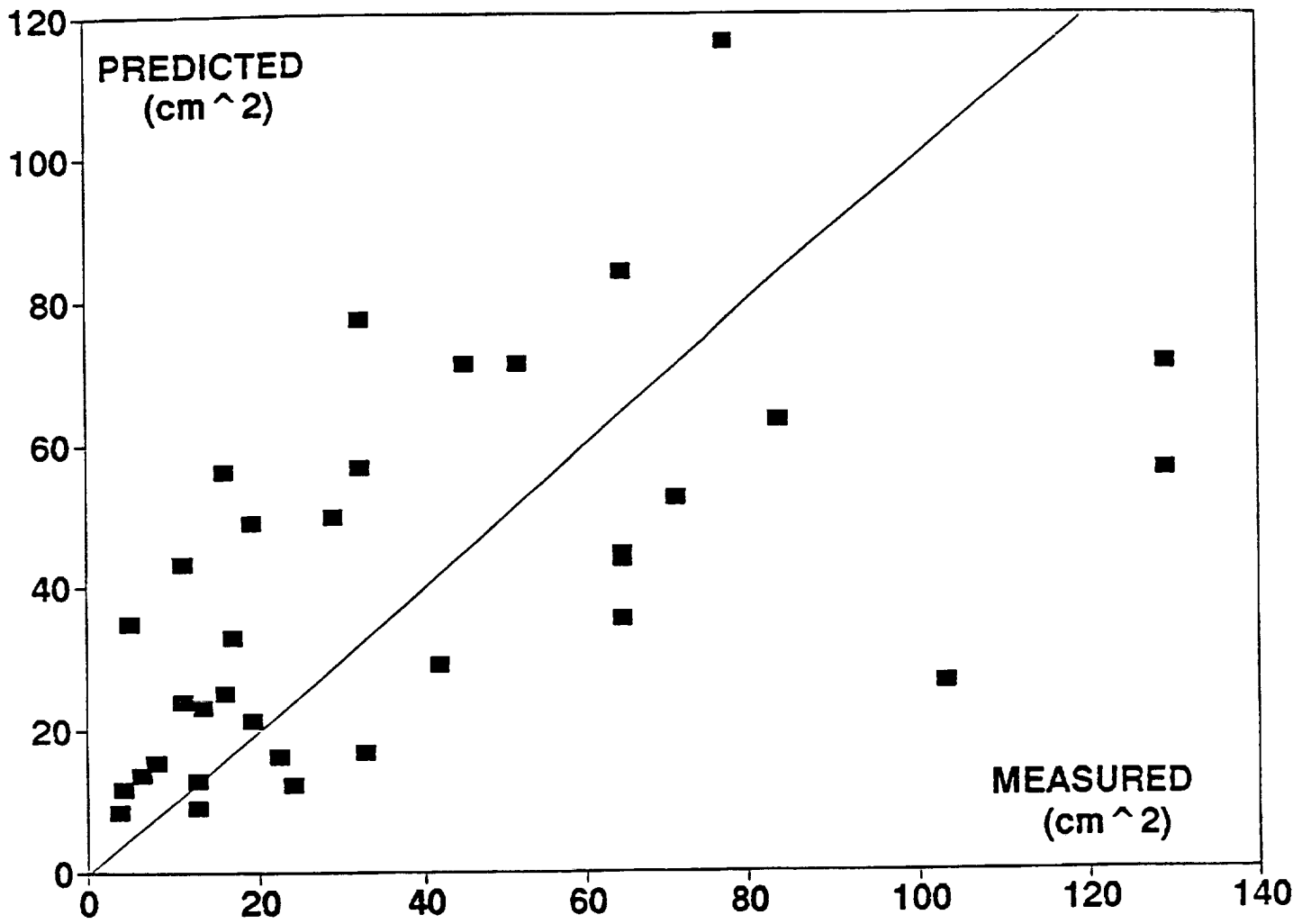


Fig. 3.8 Plot of measured VS predicted data for the MLI penetration area.

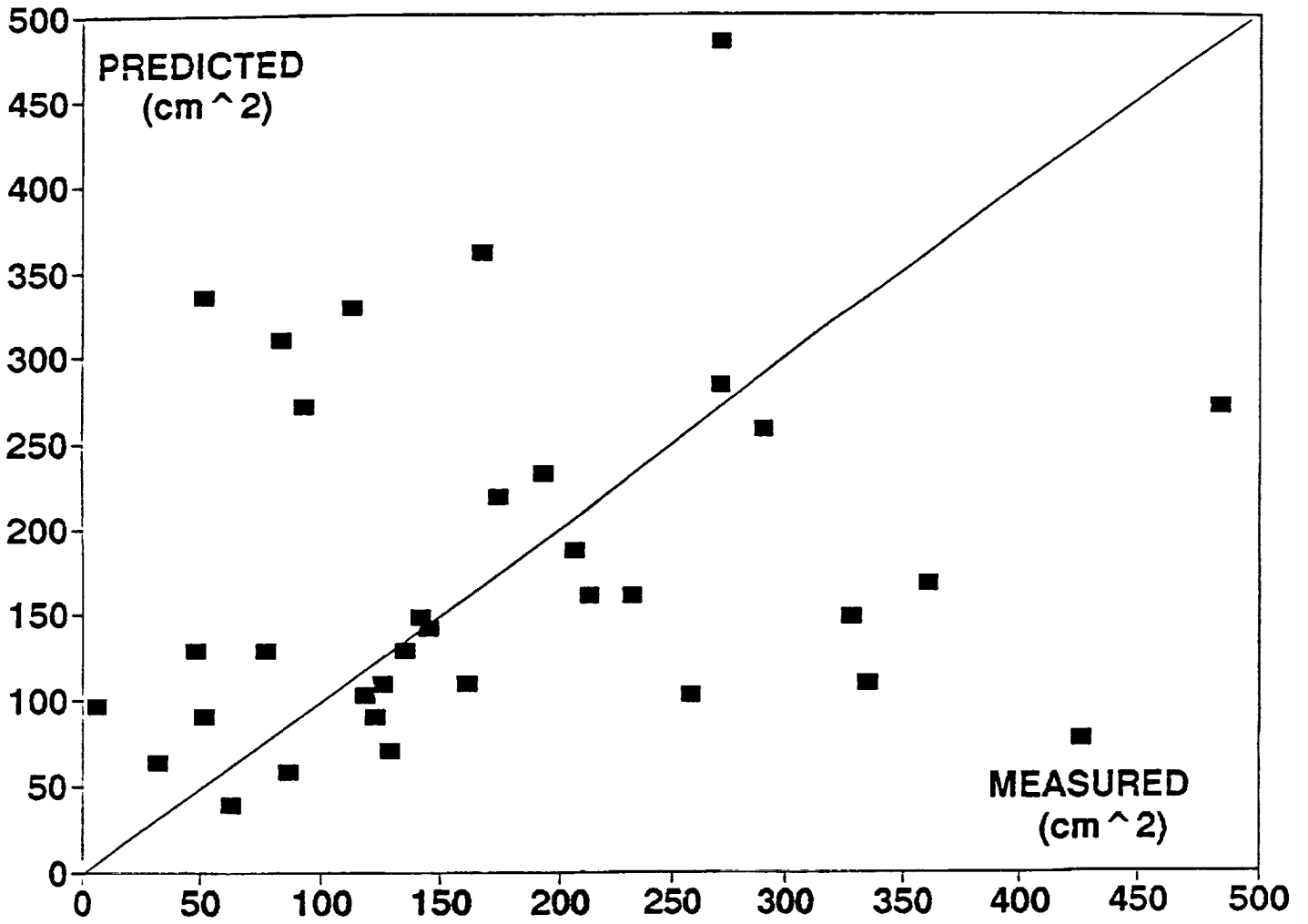


Fig. 3.9 Plot of measured VS predicted data for the MLI perforated/charred area.

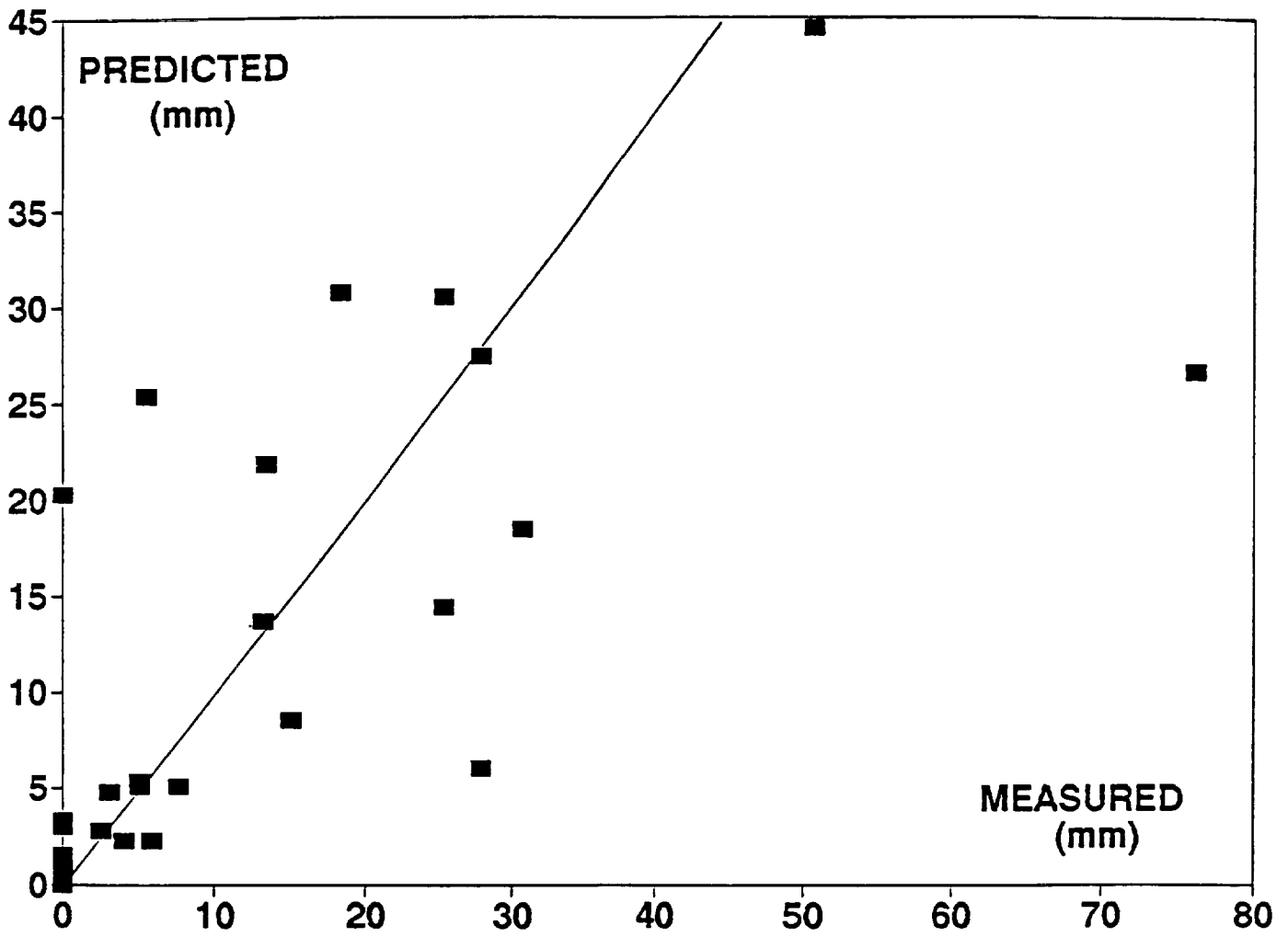


Fig. 3.10 Plot of measured VS predicted data for the major axis of the pressure wall hole.

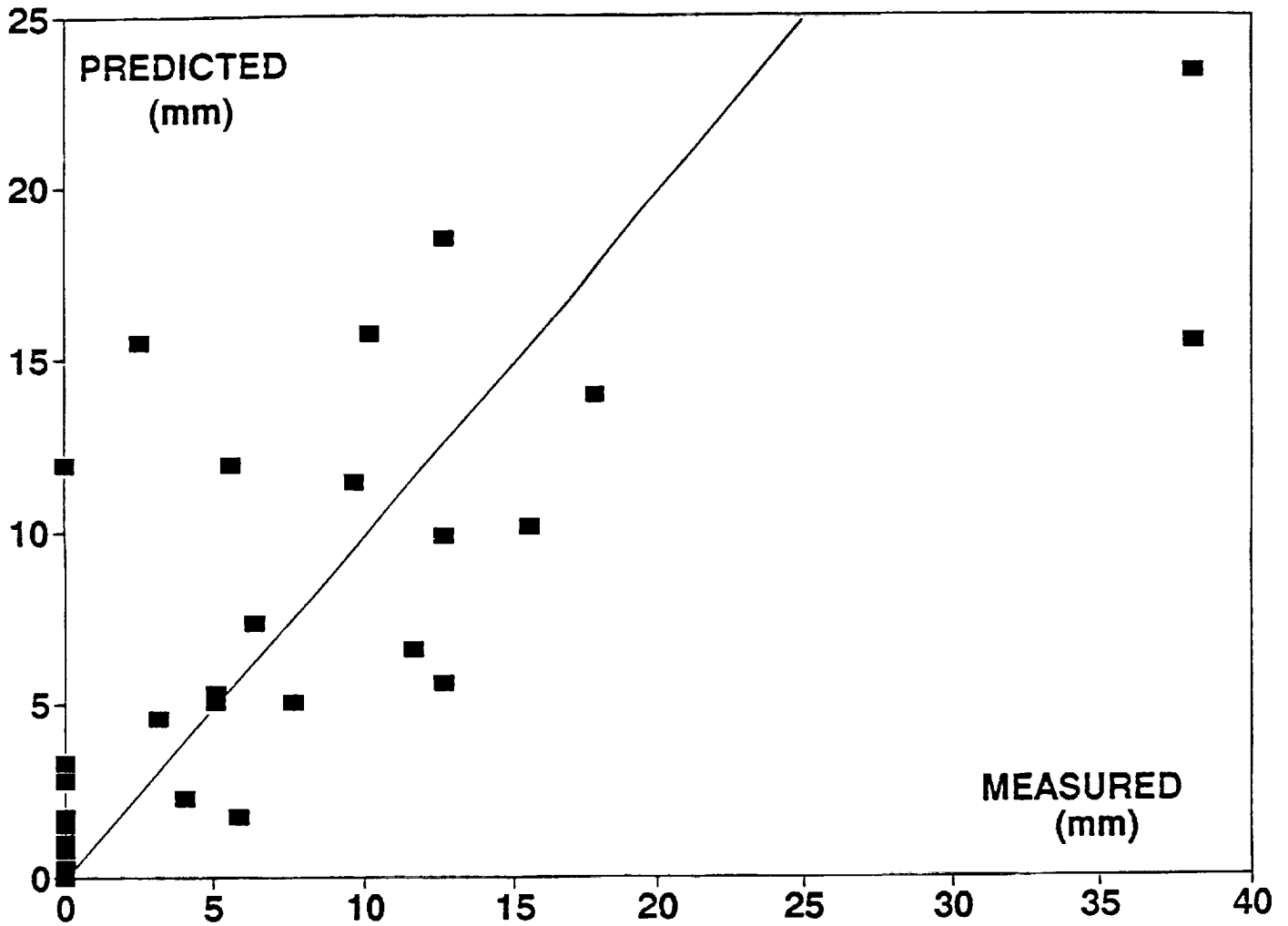


Fig. 3.11 Plot of measured VS predicted data for the minor axis of the pressure wall hole.

Table 3.2 Coefficients of Determination for Predictions

Data Set	Coefficients of Determination (y = prediction, x = measured)	
	Line of Form $y = mx + b$	Line of Form $y=x$
Bumper Major Axis	0.815	0.811
Bumper Minor Axis	0.774	0.773
MLI Penetration Area	0.322	0.289
MLI Perforated/Charred Area	0.042	-0.445
Pressure Wall Major Axis	0.541	0.538
Pressure Wall Minor Axis	0.575	0.566

4. THE POLYNOMIAL FUNCTION PREDICTION TECHNIQUE

In this section the polynomial function prediction technique is described. This method is based on the concepts associated with the finite element method (FEM). In FEM, relatively low order polynomials are used to interpolate the functions of interest (such as displacements, temperatures, and velocities) over a small portion of domain where the function is active called an element. The coefficients of the polynomial are derived from known values of the function of interest at points called nodes on the boundary of the element. For this application, the nodal values of the functions of interest (bumper hole size and so forth) were measured experimentally and are thus known quantities. This technique involves randomly selecting a sufficient number of experimental data (node) points and then determining the coefficients of the polynomial from this data.

During the course of this project, a more sophisticated polynomial interpolation approach was attempted using the isoparametric function mapping technique of FEM. This approach in its current state was not found to be suitable for engineering trade study purposes. The interested reader can consult Appendix 1 for more details on this approach.

Ideally, the nodes "closest" to the prediction point in impact parameter space should be used to evaluate the polynomial coefficients and thus make a prediction. However, the set of closest nodes may not form linearly independent set of data, making it impossible to solve for the polynomial coefficients. Also, one of the closest nodes may have a large experimental error which would contaminate the prediction. Accordingly, the computer program randomly picks subsets of data from the experimental results database file and attempts to make a prediction. If the data is linearly independent and a prediction is obtained, then the prediction magnitude and the mean distance of the data points from the prediction point are shown on the screen. The program is currently designed to seek five predictions.

In general, the impact parameters will vary greatly in magnitude. In hypervelocity impact work, dimensions can be of order 10 and velocities of order 10^6 . This polynomial function approach requires a reasonable scheme for determining "distances" between data points in impact parameter space. This is accomplished in the program by scaling the impact parameters such that their mean value is equal to unity. Of course, the dependent variables, such as bumper hole size, need not be scaled. Having scaled the independent variables, the usual formula for determining the "distance", R_1 , between two points (experimental data point and the prediction or interpolation point) in

a multidimensional space can be used:

$$R_i^2 = \sum_{j=1}^N (x_{j,i} - x_{j,INT})^2 \quad (4.1)$$

The need for scaling the independent variables is evident from considering the form of Eq. (4.1).

The form of the polynomial will now be considered. FEM theory dictates that a "complete" polynomial should produce the best results.¹⁰ Here we have four independent variables, $x_{j,i}$ ($j = 1$ to 4), associated with the i -th experimental data point to consider (bumper thickness and so forth). It was decided to use $\Delta x_{j,i}$ values in the polynomial equation to simplify the calculations. The lowest order complete polynomial for this case is:

$$\begin{aligned} D_i = & C_1 + C_2 * \Delta x_{1,i} + C_3 * \Delta x_{2,i} + C_4 * \Delta x_{3,i} + C_5 * \Delta x_{4,i} + \\ & C_6 * \Delta x_{1,i} * \Delta x_{2,i} + C_7 * \Delta x_{1,i} * \Delta x_{3,i} + C_8 * \Delta x_{1,i} * \Delta x_{4,i} + \\ & C_9 * \Delta x_{2,i} * \Delta x_{3,i} + C_{10} * \Delta x_{2,i} * \Delta x_{4,i} + C_{11} * \Delta x_{3,i} * \Delta x_{4,i} + \\ & C_{12} * \Delta x_{1,i} * \Delta x_{2,i} * \Delta x_{3,i} + C_{13} * \Delta x_{1,i} * \Delta x_{2,i} * \Delta x_{4,i} + \\ & C_{14} * \Delta x_{1,i} * \Delta x_{3,i} * \Delta x_{4,i} + C_{15} * \Delta x_{2,i} * \Delta x_{3,i} * \Delta x_{4,i} + \\ & C_{16} * \Delta x_{1,i} * \Delta x_{2,i} * \Delta x_{3,i} * \Delta x_{4,i} \end{aligned} \quad (4.2)$$

Sixteen linearly independent data points, D_i , are required to determine the sixteen polynomial coefficients, C_i . Eq. (4.2) allows for a linear variation in damage along each coordinate axis in the design space. Obviously, allowing for a quadratic variation in the damage would provide a much better fit to the data. Unfortunately, a "complete" quadratic function with four variables would require 81 linearly independent experimental data points with the MLI placement and the material types the same for all the data points. Currently, experimental data of this nature is not available.

Coefficient C_1 is the prediction of the damage at the point in the design space where the prediction is required since this is the value of the polynomial when all $\Delta x_{j,i}$ are set equal to zero. If one or more of the prediction parameters, such as bumper thickness, does not vary in the experimental database file then program POLYMETH will sense this and automatically take that variable or variables out of Eq. (4.2). If one impact

parameter does not vary, only eight polynomial coefficients need be determined and thus only eight linearly independent data points are required.

If there are less than 16 data records in the experimental database file, or if the data does not span the impact parameter design space, then the number of terms in Eq. (4.2) must be reduced if a solution is to be found. By pressing the F2 function key, the user can direct the program to seek a solution to the following "simpler" equation:

$$\begin{aligned}
 D_i = & C_1 + C_2 * \Delta x_{1,i} + C_3 * \Delta x_{2,i} + C_4 * \Delta x_{3,i} + C_5 * \Delta x_{4,i} + \\
 & C_6 * \Delta x_{1,i} * \Delta x_{2,i} + C_7 * \Delta x_{1,i} * \Delta x_{3,i} + C_8 * \Delta x_{1,i} * \Delta x_{4,i} + \\
 & C_9 * \Delta x_{2,i} * \Delta x_{3,i} + C_{10} * \Delta x_{2,i} * \Delta x_{4,i} + C_{11} * \Delta x_{3,i} * \Delta x_{4,i}
 \end{aligned} \tag{4.3}$$

Eq. (4.3) only requires 11 linearly independent data points to obtain a solution. However, Eq. (4.3) is an incomplete polynomial which is theoretically less desirable than the complete polynomial of Eq. (4.2). If solutions can not be found using Eq. (4.3), then the user may press function key F3 to request the computer to seek a solution the "simplest" possible polynomial:

$$D_i = C_1 + C_2 * \Delta x_{1,i} + C_3 * \Delta x_{2,i} + C_4 * \Delta x_{3,i} + C_5 * \Delta x_{4,i} \tag{4.4}$$

Eq. (4.4) only requires 5 linearly independent data points for a solution.

The computer program will repeatedly select random subsets of data from the experimental results database file and attempt to find a solution for the polynomial coefficients until five solutions have been found. The predictions associated with these solutions and the mean "distances" of the data points associated with the solutions are printed to the screen. A weighted average of the three solutions with the shortest mean distances is also determined and printed to the screen. A function of the form of Eq. (3.1) is used to determine the weighted average.

The weighted average should be considered the best value for the prediction. If the three predictions with the smallest mean distances are consistent then the prediction is probably a good one.

5. THE NONDIMENSIONAL PARAMETER PREDICTION TECHNIQUE.

In many applications it has been found that empirical functions are best represented in terms of nondimensional parameters. Reynolds number is an example of a nondimensional parameter that has found widespread use in empirical equations of fluid mechanics. Program NONDIMEN uses a series of empirical functions based on nondimensional parameters of the form given in Ref. 11:

BUMPER HOLE MINIMUM DIAMETER:

$$\frac{D_{MIN}}{D_P} = C_1 \left(\frac{V}{V_s} \right)^{C_2} \left(\frac{T_b}{D_P} \right)^{C_3} (\cos \phi)^{C_4} + C_5 \quad (5.1)$$

BUMPER HOLE MAXIMUM DIAMETER:

$$\frac{D_{MAX}}{D_P} = C_6 \left(\frac{V}{V_s} \right)^{C_7} \left(\frac{T_b}{D_P} \right)^{C_8} (\cos \phi)^{C_9} + C_{10} \quad (5.2)$$

MLI HOLE DIAMETER:

$$\frac{D_{MLI}}{D_P} = C_{11} \left(\frac{V}{V_s} \right)^{C_{12}} \left(\frac{T_b}{D_P} \right)^{C_{13}} \left(\frac{D_s}{D_P} \right)^{C_{14}} (\cos \phi)^{C_{15}} + C_{16} \quad (5.3)$$

PRESSURE WALL AVERAGE HOLE DIAMETER:

$$\frac{D_{PW}}{D_P} = C_{17} \left(\frac{V}{V_s} \right)^{C_{18}} \left(\frac{T_b}{D_P} \right)^{C_{19}} \left(\frac{D_s}{D_P} \right)^{C_{20}} \left(\frac{T_{PW}}{D_P} \right)^{C_{21}} (\cos \phi)^{C_{22}} + C_{23} \quad (5.4)$$

The function coefficients were determined using an optimization routine to adjust the values of the coefficients so as to maximize the coefficient of determination (R^2) of each of the functions. This approach to coefficient evaluation is suitable for any form of prediction function - linear or nonlinear. The nature of the optimization routine will now be described.

The magnitudes of the function coefficients can vary by several orders of magnitude. To avoid numerical problems it is advisable to work with percentage changes in the function coefficients. This approach also provides a simple way of controlling the amount of change in the function coefficients from one optimization iteration to the next. If the maximum allowable percentage change is too large, the optimizer could thrash back and forth around the optimum design point without ever converging to it. Alternatively,

if the maximum allowable percentage change is too small, then it could take an impractical number of iterations to get to the optimum design point, or the optimizer could get "stuck" in a local maximum of the coefficient of determination function before getting to the global maximum.

The maximum allowable change in the in the nondimensionalized design variable magnitudes is called the "search domain parameter". This is a user controlled input parameter. A value of 1.0 (equivalent to a 100% change) is recommended. The optimizer is designed to reduce the magnitude of the search domain parameter as the optimization process proceeds. The final value will be 1/100 of the initial value. The idea here is to allow large changes in the design variables initially, to quickly get into the vicinity of the global maximum in the design space, and then use finer steps to precisely locate the global maximum. The user is free to change this parameter to attempt to improve optimization efficiency.

The initial values of the function coefficients are set equal to zero. Optimal values of the function coefficients could be positive negative or zero.

The method chosen here for search vector selection is based on Powell's method¹². This is a first order method that does not require the calculation of the gradient vector. Here, Powell's method was modified as follows. Initially, a number of search vectors equal to the number of function coefficients are created. The components of these vectors are random numbers between -1 and +1. The components of each random search vector are then scaled, such that the largest component has a magnitude of unity. These vectors are stored as columns of a "search matrix". Next, the coefficient of determination is evaluated at the current point in the design space and at design points given by +/- the search domain parameter times the first column of the search matrix. If either of the + or - design points has a coefficient of determination greater than that of the current design point, then the design point corresponding to the the highest coefficient of determination will become the new design point. Otherwise, the design point does not change. The search vector multiplier (+/- search magnitude parameter or zero) used with the search vector is stored for later use. This procedure is then repeated with the remaining columns of the search matrix.

A new search vector is created after using all of the search vectors in the search matrix. This new vector is created by vectorially adding together all of the search vectors times their search vector multipliers. The new search vector is a vector sum of previous successful search vectors since

unsuccessful search vectors have search multipliers of zero. Thus, the new search vector represents (stores) the trend of the optimization process. The new search vector is scaled such that the magnitude of its largest component is unity and then is used to replace the first column of the search matrix. The procedure is repeated, a new search vector is determined, and then used to replace the second column of the search matrix, and so forth until only the last column of the search matrix remains untouched. Then an entirely new search matrix is created using the random number generator, and the process continues.

If at any time in the iterative process, a new search vector has a magnitude of zero (implying all current search directions are not beneficial), then a new random search matrix is created immediately. The random number generator uses a seed based on the number of seconds from midnight on the computer's clock. Each successive run of the optimizer will use a different set of search vectors. Currently, the program runs the optimizer three times (each time using different sets of random search vectors) to help ensure that the global maximum of the coefficient of determination has been located in the design space.

The number of search search matrices generated is governed by a user input parameter called the "iteration parameter". The number of random search matrices generated is equal to the number of design variables times the iteration parameter. The recommended value for the iteration parameter is 20.

As can be seen from the test runs of Table 5.1, the optimizer produced very consistent coefficients of determination for all four prediction equations (Eqs. 5.1-5.4). It was noted that virtually identical coefficients of determination could be produced by prediction functions having very different coefficient magnitudes as is illustrated in Table 5.2. This is a typical characteristic of nonlinear equations.

After the prediction function coefficients have been determined and displayed on the screen, the user will be prompted for the impact parameters (such as bumper thickness) associated with the desired predictions. Multiple predictions can be made from the same set of prediction coefficients.

Table 5.1 Prediction Function Coefficients of Determination for Several Runs of the Optimizer

Prediction Function	Coefficients of Determination (R^2)					Average	Std. Dev.
	Run 1	Run 2	Run 3	Run 4	Run 5		
Bumper Hole Minimum Dia.	0.9946	0.9951	0.9960	0.9945	0.9960	0.9952	0.0006
Bumper Hole Maximum Dia.	0.9949	0.9947	0.9956	0.9952	0.9956	0.9952	0.0004
MLI Hole Diameter	0.9809	0.9809	0.9805	0.9815	0.9814	0.9810	0.0004
Pressure Wall Hole Dia.	0.8292	0.8290	0.8242	0.8290	0.7740	0.8171	0.0232

Table 5.2 MLI Hole Diameter Prediction Coefficients for Several Optimizer Runs

Optimizer Runs	Prediction Function Coefficients of Eq. (5.3)						R ²
	C11	C12	C13	C14	C15	C16	
Run 1	2.860	0.463	-0.390	0.061	-0.356	0.854	0.981
Run 2	2.097	0.570	-0.472	0.042	-0.464	2.100	0.981
Run 3	1.769	0.557	-0.513	0.111	-0.430	1.781	0.981
Run 4	2.697	0.699	-0.489	-0.176	-0.615	3.401	0.982
Run 5	2.690	0.829	-0.552	-0.289	-0.767	4.043	0.981

6. A COMPARISON OF THE ACCURACY OF THE PREDICTION TECHNIQUES

The accuracies of the three prediction techniques discussed herein were compared with respect to a common impact data set, Table 6.1. This is the same data set that was recently tested for insulation damage in the Sunspot Thermal Vacuum Chamber. This data is also provided on the computer disks as experimental database file MLI.DAT. These specimens had the MLI mounted next to the bumper during impact testing. Ref. 11 contains more general details on the impact testing.

The accuracy of each prediction technique was tested by first removing a data record from the experimental database file, and then using the remaining data to make a prediction for the impact damage associated with the impact parameters of the removed data record. This was repeated for all of the 19 data records of Table 6.1. The results of this accuracy check are shown in Tables 6.2 to 6.4 for the three prediction techniques. To compare the accuracies of the three prediction techniques, average percentage differences were calculated for each of the four prediction functions. These are summarized in Table 6.5. Here, average percentage difference is the average magnitude of the difference between the predicted and measured values divided by the average magnitude of the measured values, times 100. Thus, relatively high average percentage differences indicate that the prediction function did a poor job of predicting the damage.

The following observations can be made about Table 6.5:

1. The poorest predictions by far were made for the pressure wall hole diameter.
2. The best predictions were made for the minimum bumper hole diameter.
3. The inverse R and nondimensional functions did an acceptable job for engineering trade study purposes (average percentage differences < 20%) for predicting the bumper hole size and the MLI hole diameter.
4. The nondimensional function technique did the best job overall of predicting impact damage.

The nondimensional function approach did the best job of predicting the data of Table 6.1. However, different data sets could produce significantly different results. The nondimensional function approach may not work as well if the prediction parameters (such as impact velocity) cover a greater range in the database. Also, the inverse R method has the advantage of being able to easily incorporate additional impact parameters. The other two prediction techniques are not as flexible. For instance, the inverse R method would be the method of choice for the case where different materials are used for the

Table 6.1 Experimental Data from Hypervelocity Impact Tests for Prediction Method Comparisons

Test ID	Bumper Thick.(mm)	Pr. Wall Thick.(mm)	Proj. Diam.	Impact Angle θ (deg)	Proj. Vel.(km/s)	Bump. Maj.Ax. (mm)	Bump. Min.Ax. (mm)	MLI Hol Dia. (mm)	MLI Mas Loss (grams)	Pr. Wall Maj.Ax. (mm)	Pr. Wall Min.Ax. (mm)
1012	2.03	3.18	7.95	0	6.72	18.52	18.52	55.88	0.94	15.24	3.81
1029	2.03	3.18	6.35	0	4.98	13.06	13.06	50.80	0.50	3.81	1.02
1028	1.60	3.18	6.35	0	6.98	12.95	12.95	48.26	0.83	2.54	2.54
1026	1.60	3.18	6.35	0	5.22	13.26	13.26	50.80	1.20	0.00	0.00
1018	2.03	3.18	6.35	45	6.28	18.34	14.43	48.26	0.87	8.89	8.89
1020	2.03	3.18	6.35	45	6.91	17.60	15.52	48.26	0.85	1.52	1.52
1017	2.03	3.18	6.35	45	6.20	17.93	17.93	53.34	0.88	5.59	5.59
1019	2.03	3.18	6.35	45	6.84	16.79	14.99	48.26	0.81	8.89	8.89
1027	1.60	3.18	6.35	0	7.05	13.18	13.18	48.26	0.48	4.32	4.32
1035	1.60	3.18	6.35	45	6.21	14.33	13.34	50.80	0.98	6.86	5.21
1034	1.60	3.18	6.35	45	5.20	16.54	13.16	48.26	1.02	3.30	3.30
1016	2.03	3.18	4.75	45	6.13	13.59	12.01	38.10	0.33	0.00	0.00
1023	1.60	3.18	4.75	0	4.18	10.11	10.11	27.94	0.34	0.00	0.00
1025	1.60	3.18	4.75	0	4.63	9.60	9.60	30.48	0.44	0.00	0.00
1033	1.60	3.18	4.75	45	6.28	15.04	11.76	35.56	0.53	0.00	0.00
1021	2.03	3.18	4.75	45	5.65	13.16	11.76	25.40	0.42	1.52	1.52
1032	1.60	3.18	4.75	45	4.75	12.12	10.19	25.40	0.11	0.00	0.00
1024	1.60	3.18	4.75	0	3.79	10.97	9.53	17.78	0.15	0.00	0.00
1031	1.60	3.18	6.35	0	6.98	13.69	12.73	38.10	1.05	0.00	0.00

Table 6.3 Typical Prediction Results for Polynomial Impact Damage Function

Data Set No.	Bumper Hole Minimum Dia		Bumper Hole Maximum Dia		MLI Hole Diameter		Pressure Wall Hole Diam	
	Measured (in)	ABS DIF (in)	Measured (in)	ABS DIF (in)	Measured (in)	ABS DIF (in)	Measured (in)	ABS DIF (in)
1012	0.729	0.064	0.729	0.064	2.2	4.183	0.375	0.547
1029	0.514	0.002	0.514	0.002	2	1.302	0.095	0.083
1028	0.51	0.089	0.51	0.089	1.9	2.549	0.1	0.065
1026	0.522	0.028	0.522	0.028	2	1.067	0	0.198
1018	0.568	0.066	0.722	0.072	1.9	2.081	0.35	0.047
1020	0.611	0.035	0.693	0.172	1.9	2.721	0.06	0.243
1017	0.706	0.129	0.706	0.096	2.1	1.729	0.22	0.088
1019	0.59	0.019	0.661	0.088	1.9	1.969	0.35	0.542
1027	0.519	0.094	0.519	0.094	1.9	2.272	0.17	0.077
1035	0.525	0.031	0.564	0.175	2	1.708	0.238	0.520
1034	0.518	0.102	0.651	0.155	1.9	1.881	0.13	0.128
1016	0.473	0.028	0.535	0.103	1.5	0.834	0	0.184
1023	0.398	0.105	0.398	0.105	1.1	1.076	0	0.019
1025	0.378	0.001	0.378	0.001	1.2	1.449	0	0.216
1033	0.463	0.050	0.592	0.148	1.4	1.266	0	0.041
1021	0.463	0.071	0.518	0.077	1	1.231	0.06	0.138
1032	0.401	0.010	0.477	0.164	1	1.001	0	0.081
1024	0.375	0.027	0.432	0.030	0.7	1.079	0	0.103
1031	0.501	0.028	0.539	0.066	1.5	1.893	0	0.078
Averages	0.514	0.511	0.561	0.563	1.637	1.752	0.113	0.272
Ave % Dif		10.04		16.23		27.22		158.27

Table 6.2 Typical Prediction Results for Inverse R Impact Damage Function

Data Set No.	Bumper Hole Minimum Dia		Bumper Hole Maximum Dia		MLI Hole Diameter		Pressure Wall Hole Diam	
	Measured (in)	Calc (in)	Measured (in)	ABS DIF (in)	Measured (in)	Calc (in)	Measured (in)	Calc (in)
1012	0.729	0.512	0.217	0.217	2.2	1.801	0.375	0.153
1029	0.514	0.505	0.009	0.009	2	1.741	0.095	0.043
1028	0.51	0.520	0.010	0.010	1.9	1.648	0.1	0.062
1026	0.522	0.484	0.038	0.038	2	1.638	0	0.071
1018	0.568	0.738	0.170	0.072	1.9	2.170	0.35	0.229
1020	0.611	0.599	0.012	0.172	1.9	1.926	0.06	0.382
1017	0.706	0.575	0.131	0.096	2.1	1.934	0.22	0.378
1019	0.59	0.624	0.034	0.088	1.9	1.932	0.35	0.030
1027	0.519	0.515	0.004	0.004	1.9	1.699	0.17	0.050
1035	0.525	0.553	0.028	0.175	2	1.843	0.238	0.165
1034	0.518	0.526	0.008	0.155	1.9	1.806	0.13	0.196
1016	0.473	0.470	0.003	0.103	1.5	1.050	0	0.066
1023	0.398	0.395	0.003	0.003	1.1	0.881	0	0.000
1025	0.378	0.399	0.021	0.021	1.2	1.077	0	0.000
1033	0.463	0.494	0.031	0.148	1.4	1.502	0	0.101
1021	0.463	0.478	0.015	0.077	1	1.508	0.06	0.010
1032	0.401	0.499	0.098	0.164	1	1.545	0	0.086
1024	0.375	0.397	0.022	0.035	0.7	1.107	0	0.000
1031	0.501	0.512	0.011	0.027	1.5	1.900	0	0.124
Averages	0.514	0.515			1.637	1.616	0.113	0.113
Ave % Dif			8.87	15.15				
								92.69
								15.99

Table 6.4 Typical Prediction Results for Nondimensional Impact Damage Function

Data Set No.	Bumper Hole Minimum Dia		Bumper Hole Maximum Dia		MLI Hole Diameter		Pressure Wall Hole Diam		
	Measured (in)	Calc (in)	ABS DIF (in)	Measured (in)	Calc (in)	ABS DIF (in)	Measured (in)	Calc (in)	ABS DIF (in)
1012	0.729	0.6554	0.074	0.729	0.6554	0.074	0.375	0.2864	0.089
1029	0.514	0.5382	0.024	0.514	0.5382	0.024	0.095	0.2043	0.109
1028	0.51	0.5411	0.031	0.51	0.5411	0.031	0.1	0.0664	0.034
1026	0.522	0.5374	0.015	0.522	0.5374	0.015	0	0.1447	0.145
1018	0.568	0.5972	0.029	0.722	0.6789	0.043	0.35	0.1535	0.197
1020	0.611	0.6134	0.002	0.693	0.6916	0.001	0.06	0.3431	0.283
1017	0.706	0.5772	0.129	0.706	0.6792	0.027	0.22	0.2143	0.006
1019	0.59	0.6013	0.011	0.661	0.6901	0.029	0.35	0.2109	0.139
1027	0.519	0.5381	0.019	0.519	0.5381	0.019	0.17	0.1657	0.004
1035	0.525	0.5661	0.041	0.564	0.675	0.111	0.238	0.1567	0.081
1034	0.518	0.5412	0.023	0.651	0.6217	0.029	0.13	0.2277	0.098
1016	0.473	0.4913	0.018	0.535	0.5549	0.020	0	0.0587	0.059
1023	0.398	0.3896	0.008	0.398	0.3896	0.008	0	0	0.000
1025	0.378	0.4014	0.023	0.378	0.4014	0.023	0	0	0.000
1033	0.463	0.4502	0.013	0.592	0.5082	0.084	0	0.0117	0.012
1021	0.463	0.4712	0.008	0.518	0.5535	0.035	0.06	0.0611	0.001
1032	0.401	0.4216	0.021	0.477	0.5068	0.030	0	0.0729	0.073
1024	0.375	0.3784	0.003	0.432	0.3784	0.054	0	0	0.000
1031	0.501	0.5354	0.034	0.539	0.5354	0.004	0	0.1033	0.103
Averages	0.514	0.518		0.561	0.562		0.113	0.131	
Ave % Dif			5.42			6.21			66.65
									10.88

Table 6.5 Comparison of the Accuracy of the Prediction Techniques

Method	Average Percentage Differences			
	Bumper Hole Minimum Dia.	Bumper Hole Maximum Dia.	MLI Hole Diameter	Pressure Wall Ave. Hole Dia.
Inverse R	9	15	16	93
Polynomial	10	16	27	158
Nondimensional	5	6	11	67

bumper in the same experimental results database file.

7. CONCLUSIONS AND RECOMMENDATIONS

As a result of this study the following conclusions were reached:

- There is a large amount of scatter in the hypervelocity impact damage data. It is doubtful that very high prediction accuracies can be obtained regardless of the prediction technique used.
- There is not a great deal of data available for any given set of impact conditions (such as the case with MLI against the bumper). Lack of data prevents higher order prediction functions from being used.
- The inverse R method is the most flexible prediction scheme. Any number of impact parameters and any size of database can be treated.
- The nondimensional parameter functions seem to do the best job of predicting impact damage over a relatively restricted range of impact parameters.

Based on this study the following recommendations are made:

- If possible, all three prediction techniques should be evaluated to determine the best possible prediction technique for a given data set.
- The nondimensional parameter scheme should be used to make impact predictions from data sets for which the impact parameters have a relatively small range.
- The inverse R prediction technique should be used in applications where there are a large number of impact parameters (different bumper materials in a single database file for instance) or where the impact parameters vary over a wide range.
- Numerical simulation results or approximate analytical results for high velocity (10 - 15 km/sec) should be placed in the "experimental" results database file so that realistic predictions for on orbit impacts can be made with the software.

REFERENCES

- ¹Taylor, R. A., "A Space Debris Simulation Facility for Spacecraft Materials Evaluation," SAMPE Quarterly, Vol. 18, Feb. 1987, pp. 28-34.
- ²Whipple, F. L., "Meteorites and Space Travel," Astronomical Journal, No. 1161, 1947, p. 131.
- ³Schonberg, W. P. and Taylor, R. A., "Penetration and Ricochet Phenomena in Oblique Hypervelocity Impact," AIAA Journal, Vol. 27, May 1989, pp. 639-646.
- ⁴Coronado, A. R., Gibbins, M. N., Wright, M. A. and Stern, P. H., "Space Station Integrated Wall Design and Penetration Damage Control," Boeing Aerospace Company, Seattle, WA, D180-30550-1, July 1987.
- ⁵Fraas, A. P., "Protection of Spacecraft from Meteoroids and Orbital Debris," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-9904, March 1986.
- ⁶Gehring, J. W., "Theory of Impact on Thin Targets and Shields and Correlation with Experiment," High-Velocity Impact Phenomena, Kinslow, R. (ed.), 1st ed., Academic Press, New York, 1970, pp. 117-147.
- ⁷Maiden, C. J., Gehring, J. W., and McMillan, A. R., "Investigation of Fundamental Mechanism of Damage to Thin Targets by Hypervelocity Projectiles," GM Defense Research Laboratories, Santa Barbara, CA, GM-DRL-TR-63-225, Sept. 1963.
- ⁸Bouma, D. D., Burkitt, W. C., "Multivariable Analysis of the Mechanics of Penetration of High Speed Particles," Martin Marietta Corporation, NASA CR-664, Dec. 1966.
- ⁹Özişik, M. N., HEAT TRANSFER A Basic Approach, 1st ed., McGraw-Hill Book Company, New York, 1985, pp. 593-683.
- ¹⁰Cook, R. D., Malkus, D. S., and Plesha, M. E., Concepts and Applications of Finite Element Analysis, 3rd ed., John Wiley and Sons, New York, 1989, pp. 82-83.
- ¹¹Schonberg, W. P., Bean, A. J., Darzi, K. "Hypervelocity Impact Physics,"

University of Alabama in Huntsville, Final Report for Contract No.
NAS8-36955/D.O.16, July 1990.

¹²Vanderplaats, G. N., Numerical Optimization Techniques for Engineering Design: With Applications. McGraw-Hill Book Company, New York, 1984, pp. 84-87.

APPENDIX 1 A PREDICTION TECHNIQUE BASED ON AN EXTENSION OF THE
ISOPARAMETRIC FORMULATION OF THE FINITE ELEMENT METHOD - A MASTER'S THESIS BY
MR. P. WANG.

MULTIVARIABLE INTERPOLATION AND EXTRAPOLATION BASED ON THE
ISOPARAMETRIC CONCEPT OF THE FINITE ELEMENT METHOD

by

Ping Wang

A THESIS

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

LIST OF TABLES vi

LIST OF ILLUSTRATIONSviii

CHAPTER I: INTRODUCTION: 1

 Multivariable Analysis

 Purpose of Study

CHAPTER II: MODELING BY ISOPARAMETRIC CONCEPT: 4

 Generalization of the Isoparametric Concept

 Shape Functions for N-dimensional Analysis

 Strategy of Choosing Element Nodal Values

CHAPTER III: . THE INFLUENCE OF ELEMENT DISTORTION ON CALCULATED RESULTS

 19

 Element Distortion Caused by Improper Node Numbering

 Element Distortion Caused by Geometric Irregularities

 Building A Criterion for Prediction

CHAPTER IV: NUMERICAL TESTING AND RESULTS. 46

 Sources of Data and the Testing Procedure

 Testing for 3-D Case and the Results

 Testing for 4-D Case and the Results

 Testing for 5-D Case

CHAPTER V: CONCLUSIONS AND RECOMMENDATIONS. 65

REFERENCES 67

APPENDIX 70

 Listing of the Procedure of SHPSIGN

Listing of the Procedure of NODESWAP

Listing of the Procedure of NONLNSOL

Listing of the Program of ISOMODEL

LIST OF TABLES

Table		Page
3.1.	Data list of example 1	31
3.2.	Sorting the data of Table 3.1 by scaled distances	31
3.3.	Comparison of the predicted damages in example 1, based on different elements and initial guesses	35
3.4.	Predictions of damage at (3.2,3.2) made by regular and irregular element	38
3.5.	The data in example 2 are sorted by scaled distances.	39
3.6.	Comparisons of the predicted damages in example 2, based on different elements and initial guesses	41
3.7.	Comparison of the distances from each natural coordinate solved from the nonlinear solver to the origin	44
4.1.	Listing of the experimental data of debris impact on the simulated bumper of space station	47
4.2.	Listing of geological observed data used to determine basin magnitude	48
4.3.	Listing of the prediction of major diameter by the isoparametric model and SAS vs. the measured value (3-DOF)	52

4.3.	Listing of the prediction of major diameter by the isoparametric model and SAS vs. the measured value (3-DOF)	52
4.4.	Listing of the prediction of minor diameter by the isoparametric model and SAS vs. the measured value (3-DOF)	53
4.5.	Listing of the prediction of major diameter by the isoparametric model and SAS vs. the measured value (4-DOF)	57
4.6.	Listing of the prediction of minor diameter by the isoparametric model and SAS vs. the measured value (4-DOF)	58
4.7.	Listing of the prediction of basin magnitude by the isoparametric model and SAS vs. the measured value (5-DOF)	62

LIST OF ILLUSTRATIONS

Figure	Page
2.1. A 1-DOF bar element with 2 nodes shown with respect to the physical coordinate system (X) and the natural coordinate system (ξ)	6
2.2. Bilinear element and Trilinear element defined in natural coordinate space	9
2.3. The bumper designed for shielding the pressure wall of the spacecraft from hypervelocity impact of orbital debris . .	12
2.4. A 2-D example of scattered data based on the experiment of debris impact	14
3.1. An element by proper node numbering and possible element distortions by improper node numbering	20
3.2. An example of improperly numbered element results in a nonunique mapping	21
3.3. An example of a cyclically numbered element that produces a unique mapping	23
3.4. Through the process of sorting the X-coordinate and Y-coordinate of a set of four nodes, the nodes are cyclically numbered to form a nontwisted element	25
3.5. A 3-D undistorted element is formed by cyclic numbering . .	26

3.6.	Possible element distortions caused by geometric irregularities	28
3.7.	Five possible elements formed by data points listed in Table 3.2	33
3.8.	Five possible elements formed by data points listed in Table 3.5	40
4.1.	Predictions by the isoparametric model and SAS versus observed major diameter for the 3-DOF test. Diagonal line indicates predicted equal observed.	55
4.2.	Predictions by the isoparametric model and SAS versus observed minor diameter for the 3-DOF test. Diagonal line indicates predicted equal observed.	56
4.3.	Predictions by the isoparametric model and SAS versus observed major diameter for the 4-DOF test. Diagonal line indicates predicted equal observed.	59
4.4.	Predictions by the isoparametric model and SAS versus observed minor diameter for the 4-DOF test. Diagonal line indicates predicted equal observed.	60
4.5.	Predictions by the isoparametric model and SAS versus observed basin number for the 5-DOF test. Diagonal line indicates predicted equal observed.	64

CHAPTER ONE
INTRODUCTION TO MULIVARIABLE ANALYSIS

1.1 Multivariable Analysis

Multivariable analysis is concerned with data that is a function of several independent variables, which is very important and common not only in engineering analysis but also in other fields [1]. Many multivariable analyses are derived from the viewpoint of statistical theory and have been already well developed such that they can help engineers make empirical predictions from their results. Regression analysis [2] is a technique that is commonly used to analyze experimental data in various areas of research.

Interpolation schemes can provide powerful tools for determining the relationship between dependent and independent variables. For example, interpolation of scattered 2-dimensional and 3-dimensional data using the Shepherd method, has been an important subject of CAGD (Computer Aided Geometric Design) recently [3]. Many different interpolation schemes have been devised for various types of scattered data [4]. Most of these methods such as data point triangulation and B-spline interpolation, are in essence a type of surface fitting [5]. Presently, attention is focused on how to reduce geometric discontinuities, smooth the data error and apply these procedures in

computer graphics [6]. Tadeusz Liszka has proposed a local interpolation method by using a Taylor expansion of the unknown function to reduce geometric discontinuity for those schemes that fit scattered data [7].

Due to the rapid development of computers in recent years, the finite element method (FEM) has become an important tool for the solution of engineering problems. FEM is a numerical approximation procedure based on interpolating the variables of interest over finite parts of the continuum called elements. The isoparametric formulation is one of the more important implementations of FEM, where the element coordinates and element displacements are both interpolated using the same shape functions that are defined in a natural coordinate system [8].

However, currently most of the practical applications of CAGD and FEM are focused on two dimensional or three dimensional problems, and little attention is paid on the cases with more than three dimensions or independent variables.

1.2 Purpose of Study

The original motivation of this thesis was to derive a formulation from the isoparametric concept of FEM to predict the damage to spacecraft in low earth orbit from space debris traveling at hypervelocities. Because a number of physical and mechanical properties of the debris and spacecraft are expected to be related to the damage, a multivariable analysis is thus required to make

predictions from experimental data. Although Bouma and Burkitt and several others have worked on this problem since 1963, the methods used were typically based on statistical model theory [9]. The author was interested in extending the isoparametric concept of FEM to construct a new model for use in multivariable analysis. There are many different areas of research, such as geostatistics in the field of geology [10] and biostatistics in the field of biology [11], which require multivariable analyses. The FEM based model developed here was designed to be applicable to multivariable analysis problems in general.

The following chapter reviews the basic concepts of the isoparametric implementation of FEM and then extends the concepts to problems with an arbitrary number of degrees of freedom (DOF). Chapter 3 discusses the influence of element distortion on calculated results. In Chapter 4, two sets of actual experimental data are used to test the proposed model. Here, the results of the new model are compared with those from linear regression. Conclusions and recommendations are presented in Chapter 5. Listings of the computer programs developed for this investigation are given as an Appendix.

CHAPTER TWO

MODELING USING THE ISOPARAMETRIC CONCEPT

2.1 Generalization of the Isoparametric Concept

The finite element method is basically a discretization process to partition a complicated structure or system into a finite number of small parts having simple geometric shapes [12]. These small parts are called elements. A group of elements modeling a continuum is called a mesh. Points in the continuum at the corners (and sometimes along the edges) of the elements are called nodes or nodal points. The FEM explicitly determines values for the dependent variables at the nodes. Simple functions are chosen to approximate a physical quantity over each finite element. Such assumed functions are called interpolation functions or shape functions, which are functions with unit value at one nodal point and zero value at all other nodal points. Through the use of shape functions, a relationship can be established between the coordinates of every point inside an element and the element nodal coordinates (called degrees of freedom, DOF). The principal idea of the isoparametric formulation consists in using these same shape functions to interpolate the physical quantity of interest over the element [13]. Thus, a similar relationship can be established between the physical quantities at every point inside an element, and the

element nodal physical quantities. To implement the isoparametric formulation, an orthogonal natural coordinate system is introduced such that elements described in the physical coordinate system can be mapped into an element in the natural coordinate system, where each coordinate axis varies from -1 to 1. In order to illustrate this point, a 1-DOF case is derived as follows.

Consider a bar element with two nodes which lies along the X-coordinate axis, as shown in Fig. 2.1. Because we want to have the whole element mapped from the physical coordinate X to the natural coordinates, say ξ , where $-1 \leq \xi \leq 1$, the following correspondence (boundary conditions) must occur :

1. when $\xi = -1$, $X = X_1$;
2. when $\xi = 1$, $X = X_2$

where X_1 and X_2 are the nodal X coordinates. The shape functions here must be linear in form since there is 1-DOF and only two boundary conditions. Thus, a suitable equation for writing the physical coordinate as a function of the natural coordinate is :

$$X = \frac{1}{2} (1 - \xi)X_1 + \frac{1}{2} (1 + \xi)X_2 \quad (2.1)$$

or

$$X = \sum_{i=1}^2 N_i X_i \quad (2.2)$$

where $N_1 = \frac{1}{2} (1 - \xi)$ and $N_2 = \frac{1}{2} (1 + \xi)$ are the shape functions.

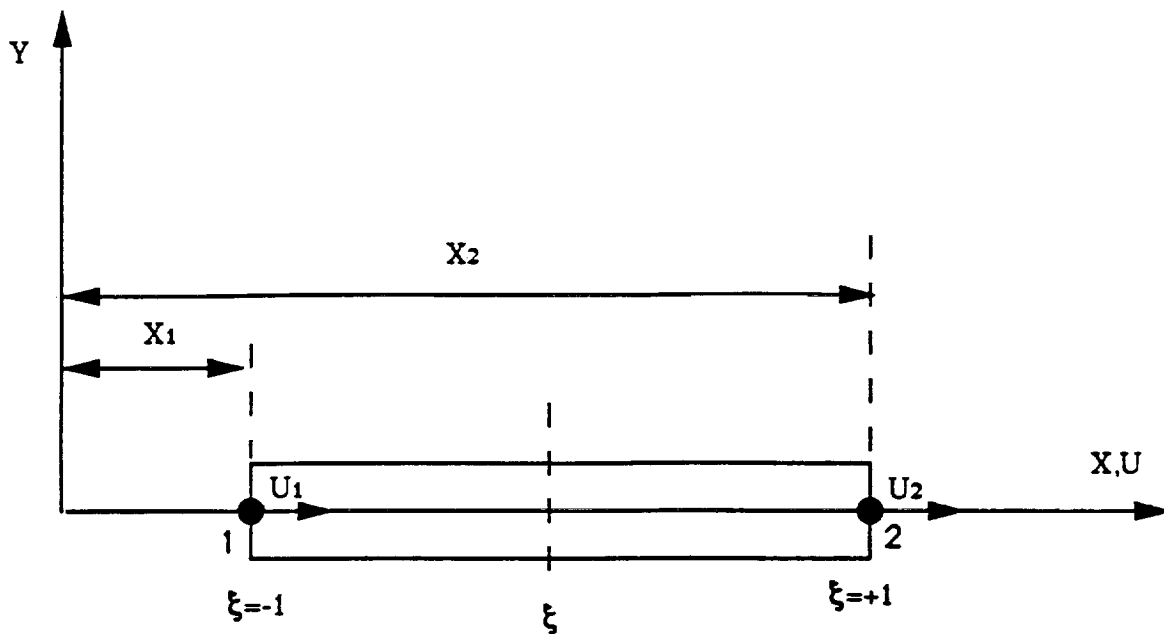


Figure 2.1 A 1-DOF bar element with 2 nodes shown with respect to the physical coordinate system (X) and the natural coordinate system (ξ).

Equation 2.1 or 2.2 establishes our desired mapping relationship. Now, if we are given some nodal physical quantity, say displacements U_1 and U_2 , for these two nodes, then based on the principal of isoparametric formulation, the displacements U at any position in the bar can be determined by the same shape functions N_1 and N_2 . That is,

$$U = \sum_{i=1}^2 N_i U_i \quad (2.3)$$

A similar derivation can be made for the 2-DOF and 3-DOF cases except that the shape functions will have different forms. More discussion on this topic follows in next section.

After understanding the basic concept of isoparametric formulation, we begin to extend this concept by generalizing the geometric coordinates to be any physical coordinates such as, say, mass velocity or concentration.

2.2 Shape Functions for N-dimensional Analyses

As it was introduced in Section 2.1, shape functions play a paramount role in relating some physical quantity within an element to the element nodal values. The form of the shape functions depends on the number of DOF of the problem and the number of nodes in the element [14]. For the case of a 1-DOF bar element, each node just had one DOF, while for the 2-DOF case such as a quadrilateral (four sided) element, each node will correspond to 2-DOF., because we need at least 2 coordinates to describe the "location" of each node in the 2-D plane.

In this thesis, we will focus on the linear interpolation in each

DOF direction over an n-DOF element. The procedure that was used for constructing the shape functions for the 1-DOF element will be generalized for use in 2-DOF, 3-DOF and n-DOF elements in the following discussion.

Starting from the 2-DOF element, we need 4 nodes to make an element such that a linear interpolation can work along the 2-DOF directions associated with the element. This element is called bilinear element, as shown in Figure 2.2-a, and the corresponding four shape functions in terms of the natural coordinates (ξ, η) can be found by using an approach similar to that which led to equation (2.1) and written as

$$N_1 = \frac{1}{4} (1-\xi) (1+\eta) \quad (2.4)$$

$$N_2 = \frac{1}{4} (1-\xi) (1-\eta) \quad (2.5)$$

$$N_3 = \frac{1}{4} (1+\xi) (1-\eta) \quad (2.6)$$

$$N_4 = \frac{1}{4} (1+\xi) (1+\eta) \quad (2.7)$$

or

$$N_i = \frac{1}{4} (1 + \xi_i \xi)(1 + \eta_i \eta) \quad (i=1..4) \quad (2.8)$$

where ξ η are defined to vary between -1 and 1 [15]. In equation (2.8), the sign of the ξ and η terms are determined by the coordinates ξ_1 and η_1 , which can be both ± 1 as long as the node number is assigned. For example, for $i = 2$, $\xi_2 = -1$ and $\eta_2 = -1$, so, $N_2 = 1/4(1 - \xi)(1 - \eta)$.

Based on the bilinear case, it is not difficult to derive the general form of the shape functions for a 3-D element with 8 nodes,

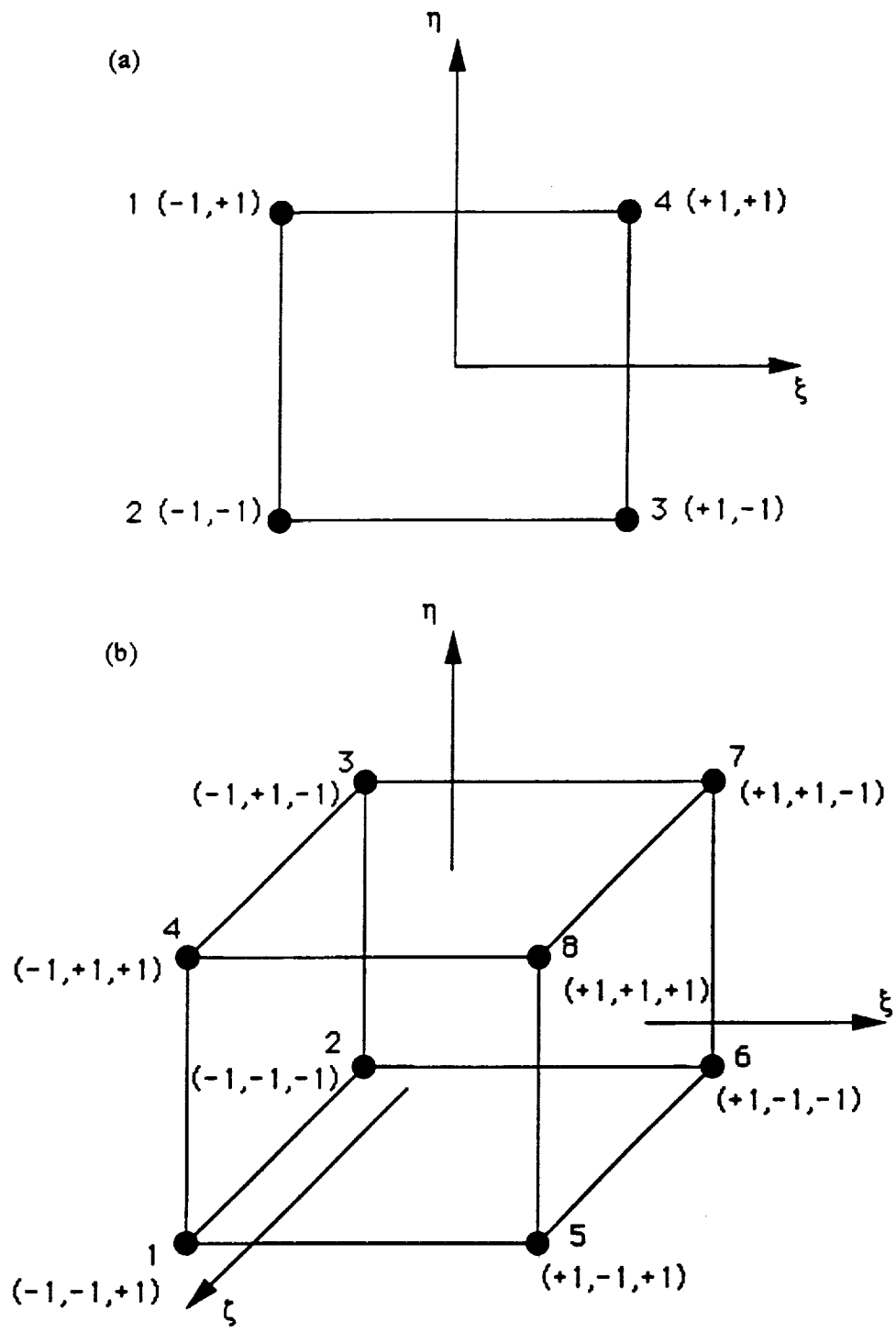


Figure 2.2 (a) Bilinear element defined in natural coordinate plane
 (b) Trilinear element defined in natural coordinate space

called a trilinear element (Figure 2.2-b). The general form is simply written as

$$N_i = \frac{1}{8}(1 + \xi_i \xi)(1 + \eta_i \eta)(1 + \zeta_i \zeta) \quad (i=1..8) \quad (2.9)$$

where ξ_i , η_i and ζ_i are equal to ± 1 when i is assigned.

In practical engineering problems, the application of FEM is limited to 2-DOF or 3-DOF elements because the physical quantities are typically assumed to be the function of spatial coordinates. If we are given a problem based on physical parameters instead of geometric coordinates, and the DOF of each node is more than 3, then a more generalized shape function is required to correlate these parameters to some physical quantity. It is not difficult to extend shape functions for 2-DOF and 3-DOF cases to apply to arbitrary n-DOF cases :

$$N_i = \prod_1^n \left(\frac{1}{2}\right)^n (1 \pm X_n) \quad (2.10)$$

n = number of DOF of each node

where each of N_i corresponds to one of 2^n "corners" of an n-DOF space, and X_n are defined to vary between -1 and 1 as before.

To this point, we have established an n-dimensional interpolation model by n-dimensional shape functions. Next, we want to apply the generalized isoparametric formulation to make multivariable interpolation for some practical problems. In order to illustrate the application of the shape functions for this objective, an example involving damage to spacecraft caused by space debris traveling at hypervelocities will be considered.

In low Earth orbit (LEO), there exists a large quantity of orbital debris that has been generated by man's activity in space over the last three decades [16]. The debris varies greatly in size — from essentially intact upper stages of rockets to small particles produced by explosions on orbit. The most dangerous particles for active spacecraft in LEO are of characteristic dimension ranging from about 0.5 mm to 2 cm because there are vast numbers of these particles, they have a high energy content, and they are too small to track by radar or other means. These particles are traveling at orbital hypervelocities (10 - 20 km/sec) and thus can inflict a significant impact damage to spacecraft. To reduce the impact damage to a minimum, Whipple [17] proposed that a protective device for a spacecraft, called a bumper, which consists of a thin aluminum outer shell or plate placed some distance from the main hull (pressure wall) of the spacecraft, Figure 2.3. The function of the bumper is to break-up or vaporize the debris particle. The pressure wall is then impacted by many tiny particles rather than a single large one. As a result, the main hull of the spacecraft sustains little damage.

In order to understand the damage that such particles will inflict upon a spaceship, an experimental approach was developed at Marshall Space Flight Center to study the damages produced in simulated spacecraft targets by projectiles fired at hypervelocities [18]. A large amount of experimental data has been collected over a wide range of impact conditions. In this thesis, the author is interested in developing a new technique for predicting spacecraft damage from the

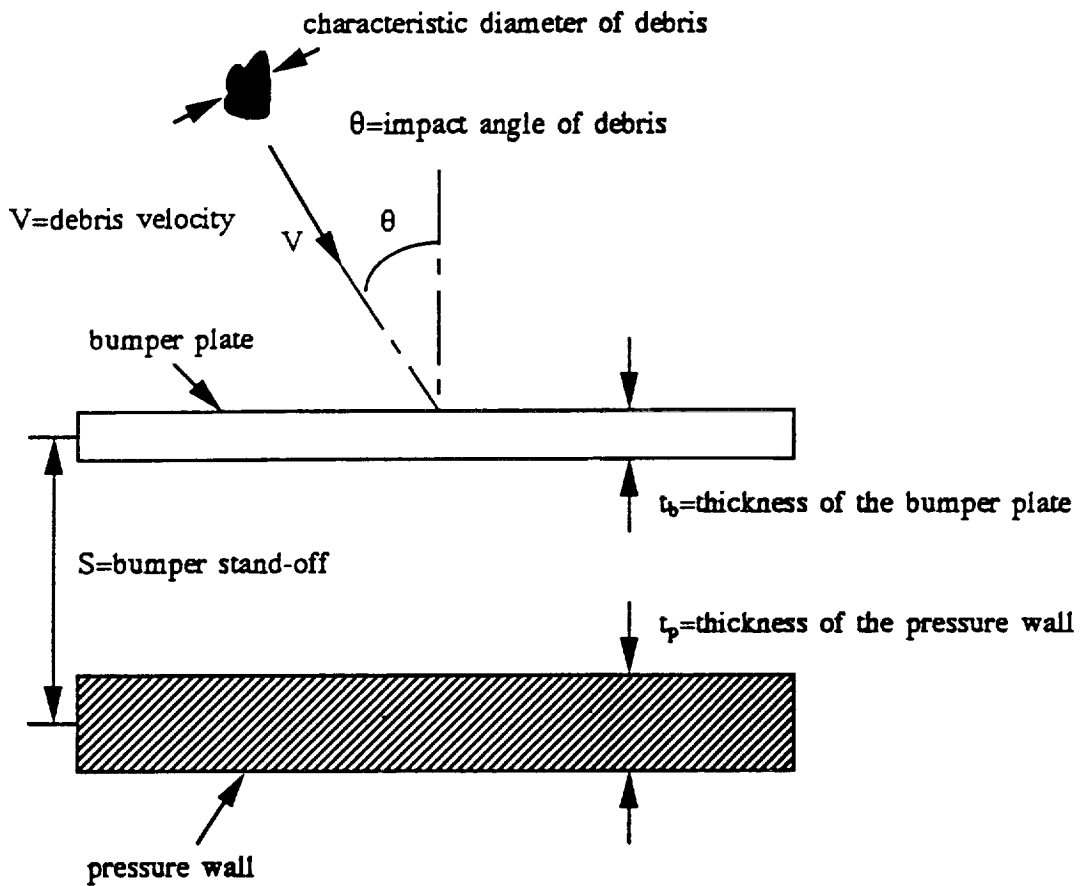


Figure 2.3 The bumper designed for shielding the pressure wall of the spacecraft from hypervelocity impact of orbital debris.

experimental results. Many attempts have been made by others to use conventional techniques to fit functions to the data with mixed success.

For illustration purposes, a simplified 2-DOF version of the impact problem will be considered first (Figure 2.4). Here it is assumed that the damage D_1 is only a function of the velocity of debris, v_1 , and the thickness of bumper, t_1 , $D_1 = D(v_1, t_1)$. D_1 could for instance be the size of the hole produced in the pressure wall by the impact. The experimental data would consist of a number of data points of the form :

$$D_1 = D (v_1, t_1)$$

$$D_2 = D (v_2, t_2)$$

$$D_3 = D (v_3, t_3)$$

.
.
.

A prediction of the damage D^* at some $v = v^*$ and $t = t^*$ is required. By the concept developed in Section 2.1, we can treat these independent parameters, v_1 and t_1 , as physical coordinates. Before we apply the shape functions, a procedure to select the most appropriate set of 4 "nodes" is required to make a prediction. These four nodes are considered to make up a finite element. This node selection procedure will be discussed in next section. Assuming the appropriate nodes have been chosen, the equations relating (v, t) in the design space to the nodal values are :

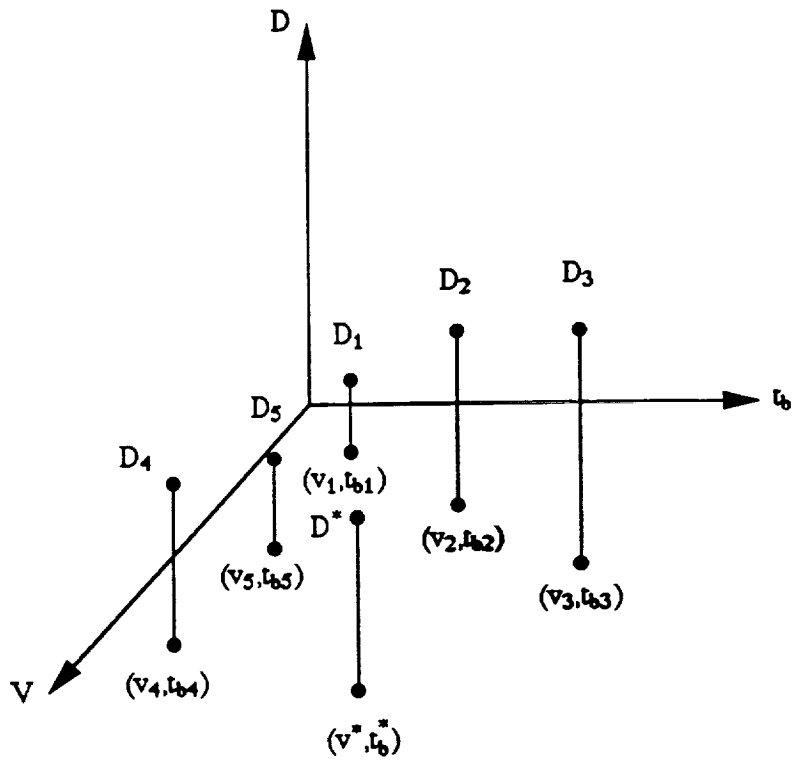


Figure 2.4 A 2-D example of scattered data based on the experiment of debris impact.

$$v = \sum_{i=1}^4 N_i v_i \quad (2.11)$$

$$t = \sum_{i=1}^4 N_i t_i \quad (2.12)$$

where v_i, t_i are known nodal values. By setting $v = v^*$ and $t = t^*$, there are two nonlinear simultaneous equations with two variables, ξ and η , to be solved because N_i are functions of natural coordinates. There are no direct methods available for solving these nonlinear simultaneous equations. Newton's method, which is based on truncating the Taylor series to only linear terms can effectively transform a nonlinear system of equations a linear system [19]. In general, an iterative approach must be used, which requires an initial guess. More details on the influence of the initial guess on uniqueness of the isoparametric mapping process will be given in next chapter. A subroutine called 'nonlinsol' was written to use Newton's method to find the roots (appendix A-2). The roots, say ξ^* and η^* , like v^* and t^* , are proposed to be related to D^* . Thus, by using the same shape functions and substituting ξ^* and η^* into the equation results in

$$D^* = \sum_{i=1}^4 N_i D_i \quad (2.13)$$

Thus a prediction for D^* is obtained.

The same procedure can be followed for the 3-DOF case. If one more parameter is added, say θ , the impact angle, then we have $D = D(v, t, \theta)$, and after choosing a set of 8 "nodes", we have

$$v = \sum_{i=1}^8 N_i v_i \quad (2.14)$$

$$t = \sum_{i=1}^8 N_i t_i \quad (2.15)$$

$$\theta = \sum_{i=1}^8 N_i \theta_i \quad (2.16)$$

By solving 3 nonlinear simultaneous equations, 3 roots will be obtained to get D^* like the 2-DOF case.

Therefore, when we have an n-DOF case, that is, we have n independent variables like v,t, θ , the same logic is repeated, but the number of DOF increases. Of course, the correspondence of N_i 's and the 2^n corners of a linear element in an n-DOF natural coordinate space is beyond the geometric imagination of the human mind. A Pascal subroutine named 'shpsign' (see appendix A-1) was written to generate the shape functions for an n-dimension element.

2.3 Strategy of Choosing Element Nodal Values

In the example of section 2.2, we outlined the basic procedure of how to perform multivariable interpolation using the isoparametric concept with the generalized shape functions. Here we discuss the problem of how to select the most appropriate nodes for a prediction.

Consider the 2-D example of $D = D(v,t)$ of the previous section. Our goal is to predict D^* , which is associated with known parameters v^* and t^* , based on the set of known D_i . Referring to Figure 2.4, we can see all the points with the coordinates of (v_i, t_i) , including (v^*, t^*) , are scattered on the v-t plane. Based on the isoparametric concept of

FEM, we need 4 nodes to make a bilinear element, such that we can interpolate D^* at (v^*, t^*) based on the 4 known nodal D_1 . However, there exists many possibilities of combining 4 sets of nodes to form a bilinear element among the scattered data. It is reasonable to choose the 4 nodes that are "closest" to (v^*, t^*) . Therefore, we have to determine all the "distances" from each (v_1, t_1) to (v^*, t^*) , and sort them by the order from the closest to farthest. Typically, the physical coordinates will vary greatly in magnitude. For the example considered here, the velocities are of order 10^3 and thickness of order 1. Thus some form of scaling is required before "distances" from point to point in the design space can be determined. The mean value of each coordinate was used as scaling factor here. Thus the distance between (v_1, t_1) and (v^*, t^*) is :

$$d_1 = \sqrt{\left[\frac{v^* - v_1}{\bar{v}}\right]^2 + \left[\frac{t^* - t_1}{\bar{t}}\right]^2} \quad (2.17)$$

where the means are given by $\bar{v} = \frac{\sum_{i=1}^m v_i}{m}$ and $\bar{t} = \frac{\sum_{i=1}^m t_i}{m}$. m is the number of sets of experimental data.

Obviously, it is easy to extend this formula to the n-D case — the number of the nodes required to form a linear element will become 2^n . Theoretically, we expect a reasonable approximation made by using the closest 2^n nodes. However, using the closest 2^n nodes will not necessarily produce the best predictions because of the possible

influence of element distortions. Therefore, on some occasions, we need to change one or two, or even all the nodes to reduce the geometric distortion of the element. This requires that different elements formed by different sets of nodes be tested. More attention will be paid on this problem in next chapter.

CHAPTER THREE

THE INFLUENCE OF ELEMENT DISTORTION ON CALCULATED RESULTS

3.1 Element Distortion Caused by Improper Node Numbering

The influence of element distortion has been an important subject in FEM, because distorted elements may produce poor results [20]. As stated in Section 2.1, the isoparametric formulation is a one-to-one mapping from a set of global cartesian coordinates to a set of local (natural) coordinates. Highly distorted elements corrupt the mapping process thereby producing unreasonable results [21]. There are two possible sources of element distortion :

- (1) distortion caused by improper elemental node numbering.
- (2) distortion caused by the geometric irregularities of the elements.

The latter source of distortion is discussed in the next section. The first type of distortion may be cured by proper node renumbering such that a non-twisted element can be obtained, as shown schematically for a 2-D case in Figure 3-1. In order to explain this, a 2-D case is considered. Figure 3.2 shows four nodes that are numbered by 1-2-4-3 in a counterclockwise (CCW) sense such that a bow-tie element is obtained. Based on the concept of isoparametric mapping, this twisted element defined in the physical coordinate system (X,Y) is mapped onto an element defined in the natural coordinate system (ξ,η) . Hence, all

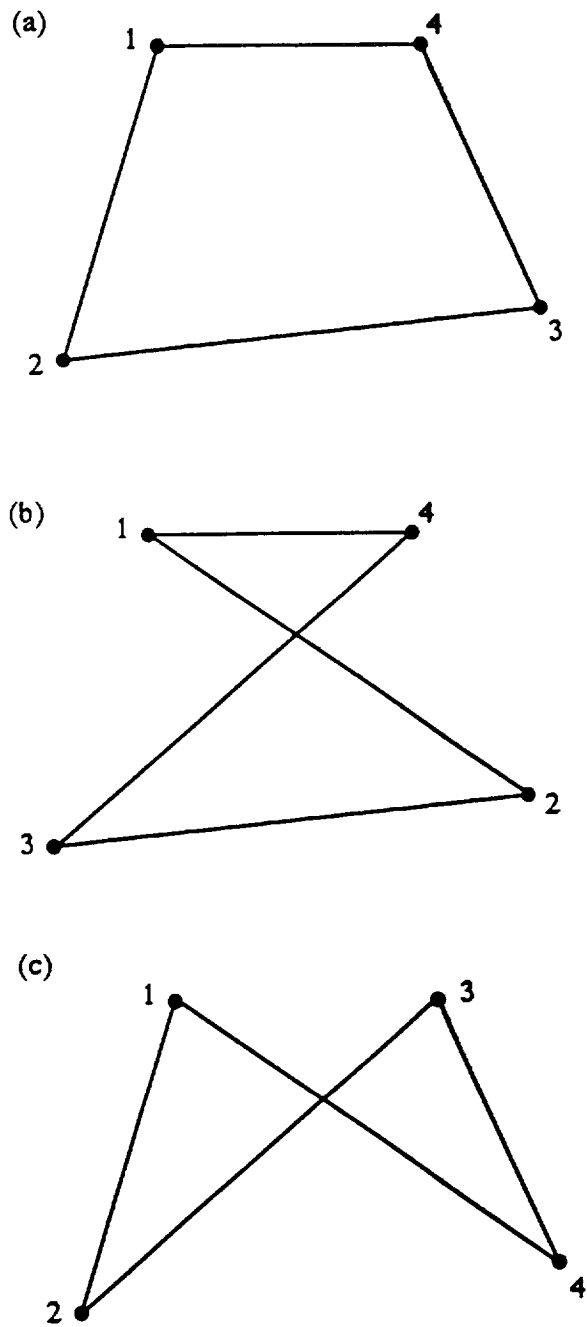


Figure 3.1 (a) An element by proper node numbering
 (b) and (c) Possible element distortions by improper node numbering

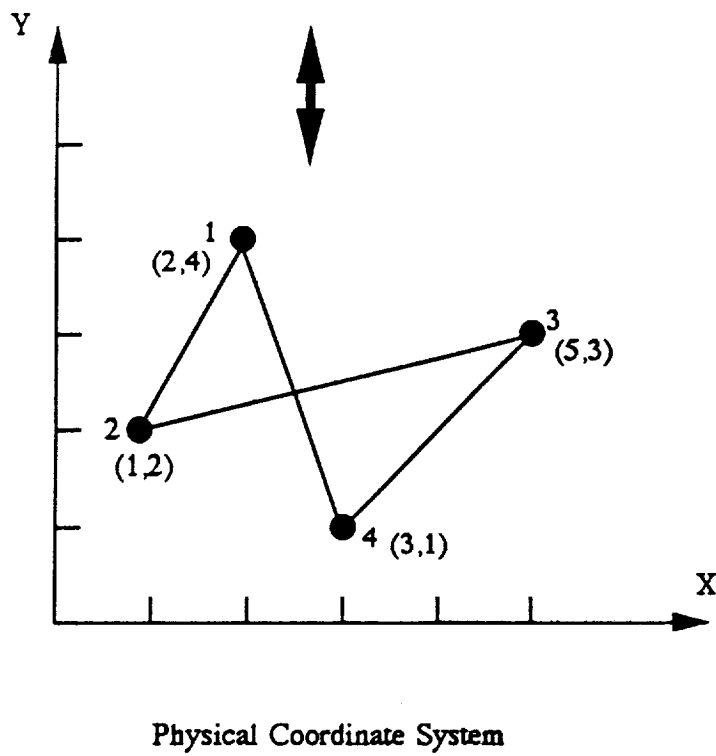
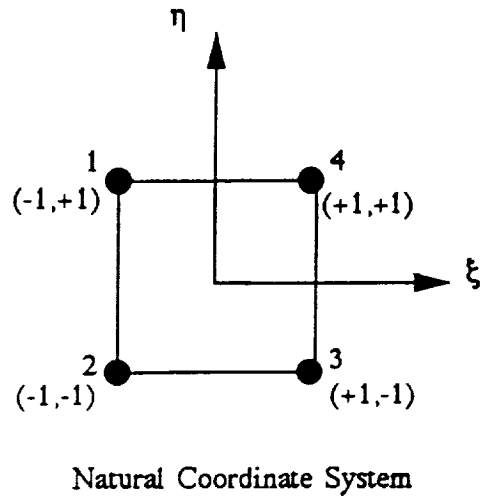


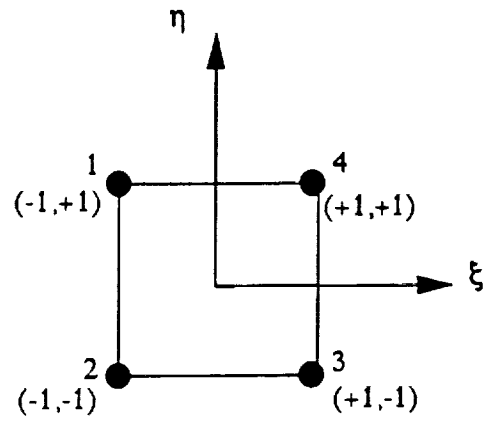
Figure 3.2 An example of improperly numbered element results in a nonunique mapping.

the points, including the four corners, inside the twisted element should be uniquely mapped onto the points of a square element in ξ - η plane. It is easy to investigate the uniqueness of the mapping by checking if the corners of the twisted element uniquely correspond to those of the square element by the node number. For example, the coordinates of node 1 in X-Y plane is (2,4), which is supposed to correspond to the coordinates of (-1,1) of the node 1 in ξ - η plane, and so on. Now, consider corner 3, whose coordinates in X-Y is (5,3) and the corresponding coordinates of node 3 in the ξ - η plane should be (1,-1). By recalling the equations (2.11) and (2.12), two simultaneous equations are obtained :

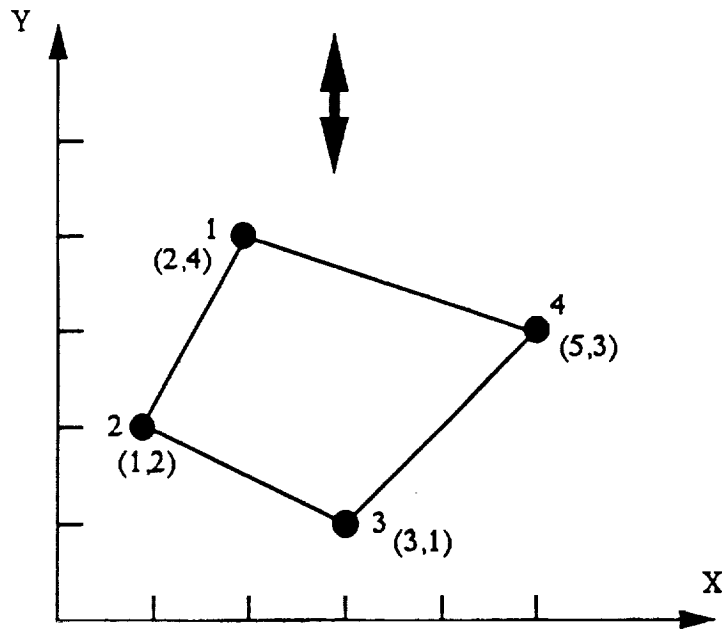
$$5 = \sum_{i=1}^4 N_i X_i$$

$$3 = \sum_{i=1}^4 N_i Y_i$$

where N_i , $i = 1..4$ are given by equations (2.4) to (2.7), and X_i and Y_i are the X and Y coordinates of the i th node, respectively. After solving these two nonlinear equations with the initial guess of $(\xi, \eta) = (0,0)$, a set of roots is obtained as $\xi = 7.0$, $\eta = 0.08$, which is obviously different from the required values of (1,-1). On the other hand, if the four nodes are numbered in a cyclic way (1-2-3-4, CCW), then a nontwisted element is obtained (Figure 3.3). Here, node 4 has (x,y) coordinates (5,3) and should have (ξ, η) coordinates (1,1). Again, by substituting these corners into equations 2.11 and 2.12 and solving the nonlinear equations with the initial guess of (0,0), we do



Natural Coordinate System



Physical Coordinate System

Figure 3.3 An example of a cyclically numbered element that produces a unique mapping.

get the root of (1,1) as was expected. The other three corners of this element are also mapped to appropriate (ξ, η) values. Thus, care must be taken to avoid element distortion caused by improper node numbering.

A systematic algorithm for determining the proper node numbering for a given element will now be discussed. Suppose a set of four nodes are given to form an element in X-Y plane (Figure 3.4-a), which must be mapped onto an element in ξ - η plane, whose four corners are already cyclically numbered 1-2-3-4, (Figure 3.4-d). First, these four nodes are partitioned into two groups by sorting their X-coordinates from the smallest to the largest, such that the first group contains two nodes with the smallest X-coordinates and the other group contains the remaining two nodes with the larger X-coordinates. In the first group, the node with the least X-coordinate is numbered 1, and the other one is numbered 2 ; in the second group, the node with smaller X-coordinate is numbered 3 and the other one is numbered 4 (Figure 3.4-b). Next, each group is separately sorted according to the Y coordinates of its two nodes. In the first group, the node with the largest Y-coordinate is numbered 1 and the other one is thus 2. Similarly, in the second group the node with largest Y-coordinate is numbered 4 and the other one is numbered 3 (Figure 3.4-c). After these two sortings, a set of cyclically-numbered nodes (1-2-3-4) is obtained (Figure 3.4-d). We can extend this sorting scheme to a 3-D case, where there will be 8 nodes to be numbered to make a least distorted hexahedral isoparametric element, Figure 3.5. As was done for the 2-D case, first the nodes are sorted with respect to their X components. Then, we partition these

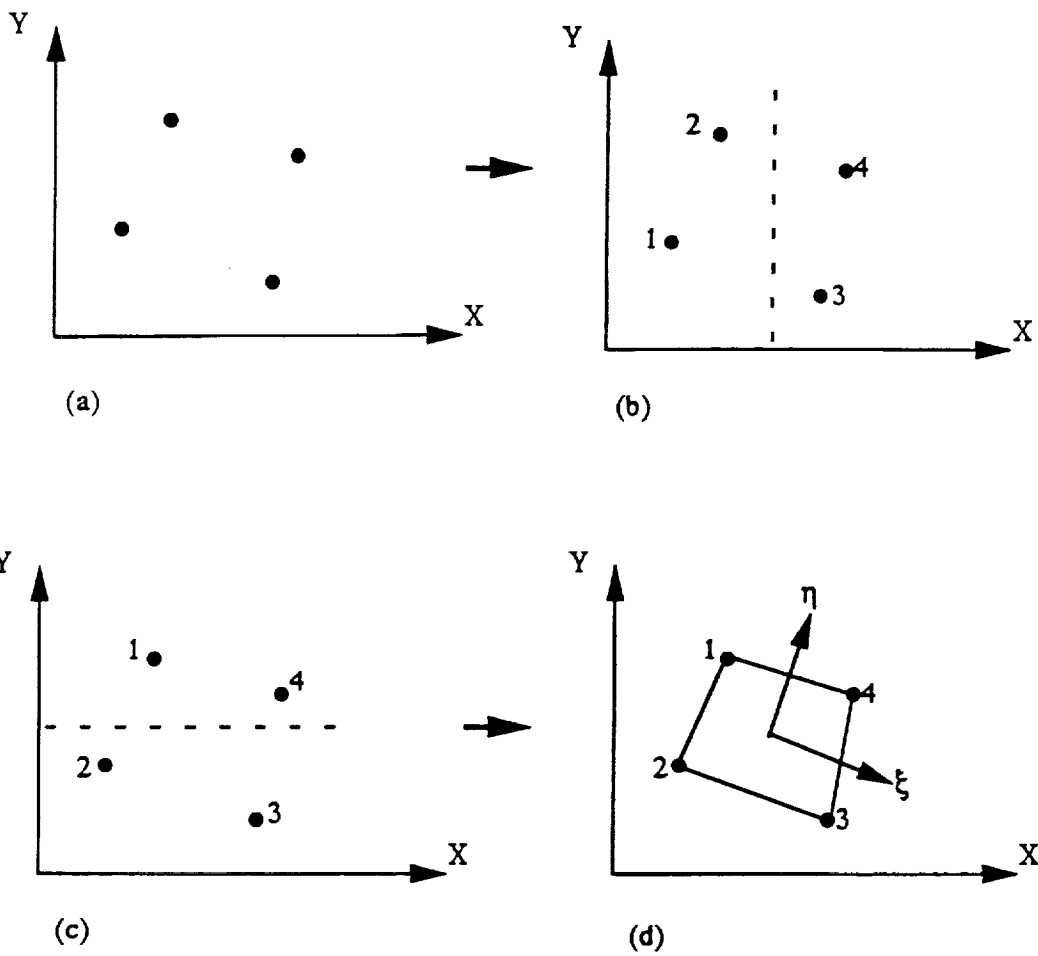


Figure 3.4 Through the process of sorting the X -coordinate and Y -coordinate of a set of four nodes, the nodes are cyclically numbered to form a nontwisted element.

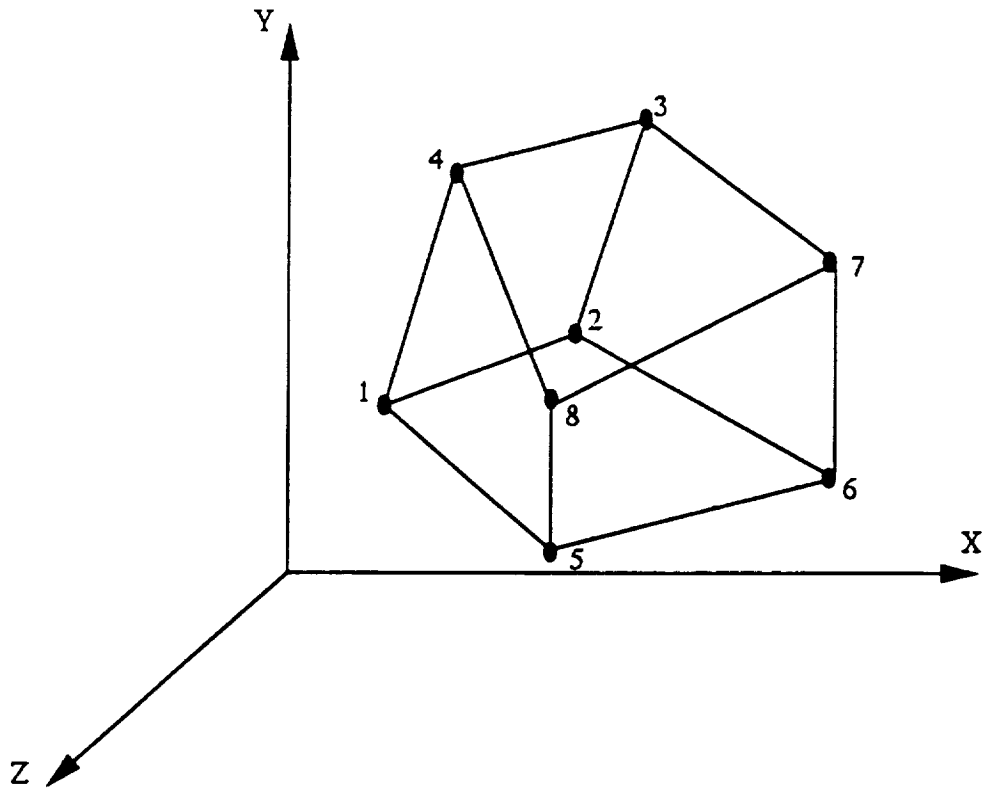


Figure 3.5 A 3-D undistorted element is formed by cyclic numbering.

nodes into two 4-node groups by the temporary order 1-4 and 5-8. Again we treat each group as a 2-D case, and obtain a second temporary order for each group by sorting their Y-components. At last, a third sorting for Z-components is conducted such that $(\text{node } 1)_z > (\text{node } 2)_z$, $(\text{node } 4)_z < (\text{node } 3)_z$ for the group of 1-4, and $(\text{node } 5)_z > (\text{node } 6)_z$, $(\text{node } 8)_z < (\text{node } 7)_z$ for the group of 5-8, so a normal element is obtained by the last order.

The same node sorting technique can be applied to problems with more than three dimensions. A pascal subroutine named "nodeswap" is written for this purpose (see Appendix 3).

3.2 Element Distortion Caused by Geometric Irregularities

In last section, minimizing element distortion by selecting an appropriate cyclic node numbering scheme was discussed. However, there are often geometric irregularities which cannot be removed by proper node numbering, Figure 3.6. These geometric defects may also cause a nonunique isoparametric mapping to occur.

In the FEM isoparametric formulation, the shape functions are always used to correlate the element nodal values to the values within the element. However, it is possible for our purpose here that the points to be predicted are outside the elements, so that the use of extrapolation is required. In this case, element distortion could make the errors that are inherently associated with extrapolation worse.

Due to these two considerations, different elements, like different meshes in FEM, are tried to find the elements that are the

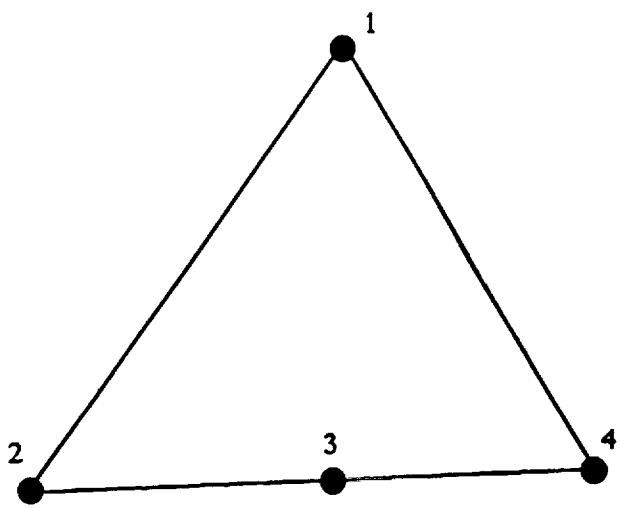
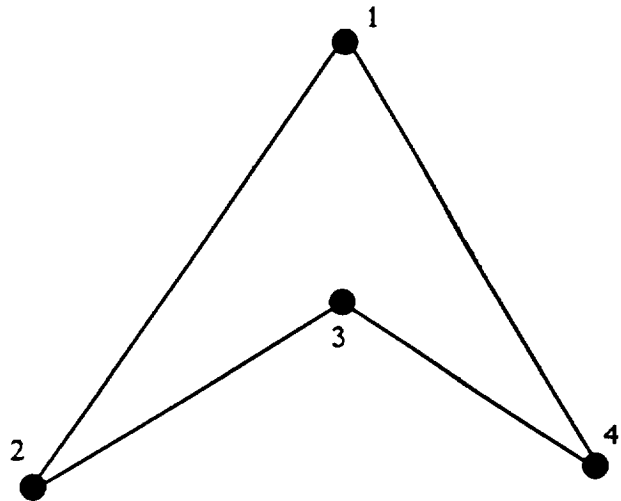


Figure 3.6 Possible element distortions caused by geometric irregularities.

least distorted and that contain the point in space where the prediction is required. Thus, in addition to the element formed by the closest 2^n nodes, we can try other possible elements formed by arbitrarily taking 2^n nodes from the first closest $2^n + 1$ nodes to make distinct combinations, from which $2^n + 1$ different elements will be generated. Similarly, elements based on the combinations of the closest $2^n + 2$, etc. nodes also can be further tested.

Next, how the geometric irregularities influence the uniqueness of the mapping will be considered. The mapping from physical coordinate space to natural coordinate space requires the solution of nonlinear simultaneous equations. In general, nonlinear equations can have more than one set of real roots, which can be found by starting from different initial guesses when Newton's method is used. If the element has just a small amount of distortion, then one set of roots (ξ, η) must uniquely exist inside or on the element borders defined by $\xi = \pm 1$ and $\eta = \pm 1$. However, it was found that some highly distorted elements still could give a unique mapping for some regions within the elements. If the point is located in the vicinity of the less distorted part of the element, then the mapping of this point could be unique. In order to explain this, an example will be considered in Section 3.2.1.

If the point to be predicted is outside all possible elements, then the natural coordinate roots must be outside the elements as well. If the linear interpolation functions are assumed to be valid outside the elements when the point of interest is sufficiently close to the

elements, then the same concept just discussed for treating a point inside a distorted element is extended to this case. That is, the uniqueness of mapping (or the roots of ξ, η) depends on if the point of interest is located in the vicinity of the undistorted part of the element. An example in Section 3.2.2. will be used to explain this application.

3.2.1 Example 1

In this section, the 2-D "debris impact" example, which was introduced in Chapter 2, will again be considered. As shown in Table 3.1, there are five nodes defined in the velocity-thickness ($v-t$) coordinate plane, where v refers to the velocity of debris and t is the thickness of bumper. Each of the nodes has an associated impact damage D , arbitrarily defined for illustration purposes by the function:

$$D(v,t) = v^2t + v + t \quad (3.1)$$

A prediction of the damage will be made at $(v,t) = (3,3)$ using the proposed method with the data of Table 3.1. Prediction will be made using five different elements and the results compared with the "exact" answer given by equation 3.1.

Now, we begin the analysis with finding the "distances" from each node to the node $(3,3)$ based on the scaling scheme introduced in chapter 2, and then sorting them by the order from the closest to the farthest, which is listed in Table 3.2. Next, we arbitrarily select 4

node	v	t	D
1	1	2	4
2	2.5	2.5	20.625
3	4	2	38
4	3	4	43
5	2	1	7
*	3	3	?

Table 3.1 Data list of example 1

no.	v	t	D	scaled distance from (v,t)=(3,3)
1	2.5	2.5	20.625	0.295
2	3	4	43	0.434
3	4	2	38	0.591
4	1	2	4	0.911
5	2	1	7	0.957

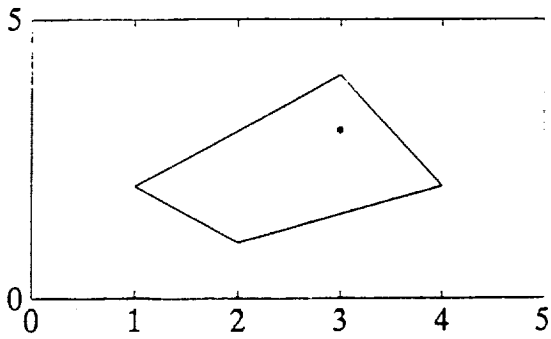
Table 3.2 Sorting the data of Table 3.1 by scaled distances

sets of nodes from these 5 nodes to make 5 different elements, of which the corners are cyclically numbered and then mapped to a natural coordinate plane. Each of these elements is separately discussed as follows:

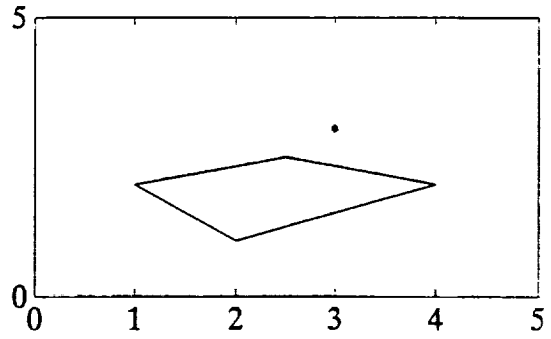
Element 1 is formed by the nodes 2, 3, 4, and 5 in Table 3.2. After cyclically numbering the nodes, the shape of the element is shown in Figure 3.7-a. Obviously, the node (3,3), marked with '*', is inside the element. By solving the nonlinear simultaneous system for mapping this node to the natural coordinate the unique root $(\xi, \eta) = (0.65, 0.29)$ is obtained whenever initial guesses between -1 and 1 are provided to the equation solver. If the initial guess is exactly one of the corners of the element in natural coordinate plane, except the corner (1,-1) which results in the roots outside the range of -1 and 1, the roots obtained are also (0.65, 0.29). However, we are just concerned with the uniqueness of the roots inside the element, so the roots outside the element will be ignored. At last, we substitute the roots of (0.65, 0.29) to equation (2.13) to determine the damage.

Element 2 is formed by the nodes 1, 3, 4, and 5 in Table 3.2. The same approach is repeated as in element 1, but the node (3,3) is outside the element as shown in Figure 3.7-b. In the process of solving nonlinear system, no matter what the initial guesses are between -1 and 1 or the corners, the root we obtain are consistently (3,2). By substituting this root to equation (2.13), a damage is predicted.

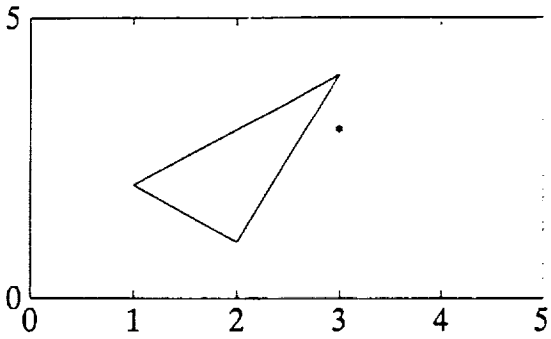
Element 3 is formed by the nodes 1, 2, 4, and 5 in table 3.2. As



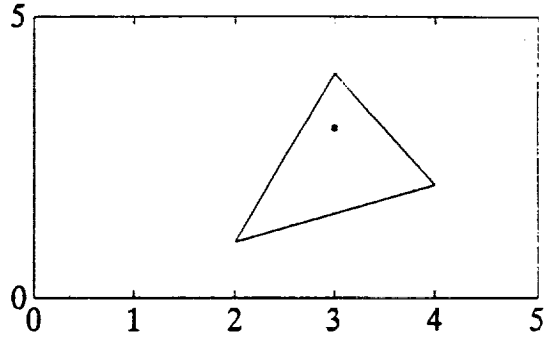
(a)



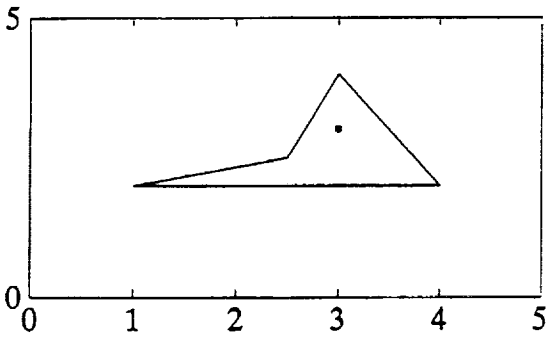
(b)



(c)



(d)



(e)

Figure 3.7 Five possible elements formed by data points listed in Table 3.2

shown in Figure 3.7-c, the node (3,3) is still outside the element, and the nodes 1, 2, and 5 are co-linear. No convergent roots can be obtained whatever initial guesses are chosen because of the serious geometric irregularity of the element. Thus, the damage cannot be correctly calculated using this element.

Element 4 is formed by the nodes 1, 2, 3, and 5 in table 3.2. As shown in Figure 3.7-d, this is also a triangular element with the nodes 1, 2, and 5 collinear, but here the node (3,3) is inside the element. Except the initial guess by the corner of (-1,1), which causes a singular matrix in Newton's method and fails to solve the roots, the other guesses uniquely generate (0.38,0.42). Based on this roots, the damage can be calculated in spite of the geometric irregularity of the element.

Element 5 is formed by the nodes 1, 2, 3, and 4 in table 3.2. As shown in Figure 3.7-e, it is also an irregular element that contains the node (3,3). The mapping is not unique because the roots are (0.48, 0.24) when the initial guesses are taken from in the area specified by $0 \leq \xi \leq 1$ and $-1 \leq \eta \leq 1$, but the roots are (-0.7,1.8) when the initial guesses are chosen from the area specified by $-1 \leq \xi < 0$ and $-1 \leq \eta \leq 1$. Thus damage predictions cannot be made using this element.

Damage prediction based on five elements discussed previously are given in Table 3.3. The prediction made by the first element should be most convincing because the element is regular and the node predicted is inside the element. The prediction value of 34.8 agrees quite well with the exact value of 33, considering that the nodal data values by

element	roots [*]	roots ^{**}	damage [*]	damage ^{**}
2345	0.65, 0.29	0.65, 0.29	3.48E+01	3.48E+01
1345	3.0 , 2.0	3.0 , 2.0	2.14E+01	2.14E+01
1245	--	--	--	--
1235	0.38, 0.42	--	3.38E+01	--
1234	0.48, 0.24	- 0.7 , 1.75	3.41E+01	2.96E+01

note:

root^{*} : root (ξ, η) obtained by initial guess of (0,0)

root^{**} : root (ξ, η) obtained by initial guess of closest corner

damage^{*} : damage based on root^{*}

damage^{**} : damage based on root^{**}

exact damage : 33

Table 3.3 Comparison of the predicted damages of example 1, based on different elements and initial guesses

an order of magnitudes (4 to 43). A discussion on the results of element 2 and element 3 will be given after the second example is considered. Here, two observations based on the results of element 1, element 4 and element 5 are given.

- (1) Elements containing the predicted node are not necessarily associated with accurate predicted results :

Based on providing the equation solver with the initial guess of (0,0), it is easy to check that elements 1, 4, and 5 contain the predicted node (3,3). However, not all the three elements guarantee a unique mapping, which can be seen when other initial guesses are used. Element 4 and 5 fail to give reasonable predictions because the uniqueness of mapping does not exist for them. Hence, an accurate prediction depends on to the uniqueness of mapping if the predicted point is known to be inside the element.

- (2) Making predictions based on different elements:

It is apparent that element 1 can make a better prediction due to its relatively undistorted shape which insures the uniqueness of the natural coordinate roots inside the element. However, it is difficult to determine the geometric irregularity of elements by trying all possible initial guesses to test the uniqueness of the mapping. Also, uniqueness mapping could occur in highly distorted elements if the point to be predicted is located in the less distorted parts of the elements. The following approach was used to cope with this problem. First, the corner closest to the point to be predicted is located for each element in the physical coordinate system. When an isoparametric

mapping is performed, the whole element in X-Y plane is mapped onto an element in ξ - η plane, where the relative position of the predicted point and the closest corner does not change. A set of roots is solved for from the nonlinear system using this corresponding "closest corner" in ξ - η plane as an initial guess. The roots (ξ, η) are then found again using $(0,0)$ the initial guess. If a root can be uniquely obtained inside the element by the initial guesses of $(0,0)$ and the "closest corner", then the point to be predicted is said to be located in a sufficiently undistorted part of the element. If the root obtained by initial guess of $(0,0)$ is different from that by the closest corner, then it implies there is an unacceptable geometric irregularity at the associated corner with reference to the physical coordinate plane. Considering element 4, the uniqueness of the roots seems valid for the guesses inside the element, but is ruined when the closest corner $(-1,1)$ is used as a guess. This is because the corner $(2.5,2.5)$ in physical coordinates, which is associated with the corner $(-1,1)$ in natural coordinate, is collinear with the other 2 corners of the element. Similarly, the element 5 is also irregular at the corner $(2.5,2.5)$, so that the associated closest corner $(-1,1)$ will lead to a different set of roots from that by the guess of $(0,0)$.

In the same example, if we try to predict the damage at $(3.2,3.2)$ just by the element 1, element 4 and element 5, we find not only the regular element 1 but also the irregular elements 4 and 5 can lead to a close approximation with the closest corners used as initial guesses, Table 3.4. This is because $(3.2,3.2)$ is much closer to these regular

element	roots [*]	roots ^{**}	damage [*]	damage ^{**}
2345	0.86,0.32	0.86,0.32	3.87E+01	3.87E+01
1235	0.76,0.36	0.76,0.36	3.84E+01	3.84E+01
1234	0.78,0.31	0.78,0.31	3.84E+01	3.84E+01

note:

root^{*} : root (ξ, η) obtained by initial guess of (0,0)

root^{**} : root (ξ, η) obtained by initial guess of closest corner

damage^{*} : damage based on root^{*}

damage^{**} : damage based on root^{**}

exact damage : 39.168

Table 3.4 Predictions of the damage at (3.2,3.2) made by regular and irregular elements.

corners than to the distorted corners.

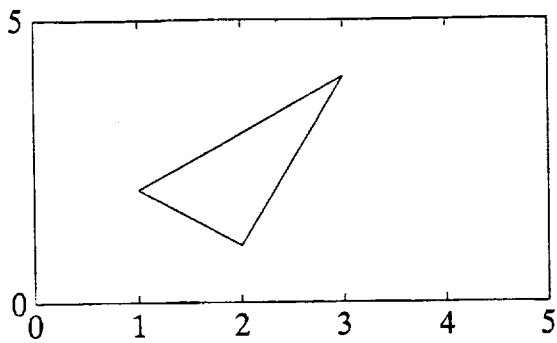
3.2.2 Example 2

This second example is presented to explain how to make a prediction when the predicted node is outside all possible elements. Suppose we are to predict the damage at (4,3) instead of (3,3), and the rest nodal data are the same as those in example 1. After finding the "scaled distances" and sorting, five different elements (see Figure 3.8 a-e) are formed by taking any four distinct nodes from the Table 3-5.

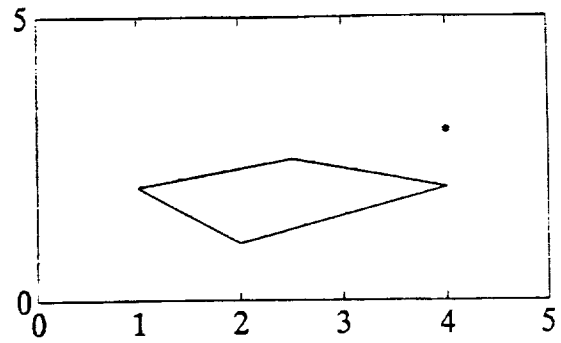
no.	v	t	D	scaled distance from (v,t)=(4,3)
1	4	2	38	0.435
2	3	4	43	0.591
3	2.5	2.5	20.625	0.638
4	2	1	7	1.182
5	1	2	4	1.276

Table 3.5 The data in example 2 are sorted by scaled distances.

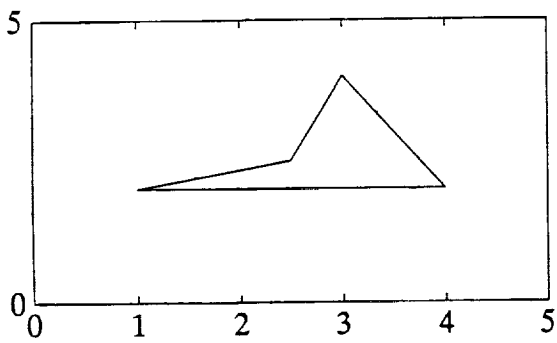
Then by using the observation made on the results of example 1 that the predicted node should be inside the element if an initial guess of (0,0) is used for solving the nonlinear system such that the root is between -1 and 1, we find the node (4,3) is actually outside these 5 elements (see Table 3.6 a-e). If linear interpolation functions are assumed applicable when the point to be predicted is outside of the element but is still sufficiently close to the element, then we can try



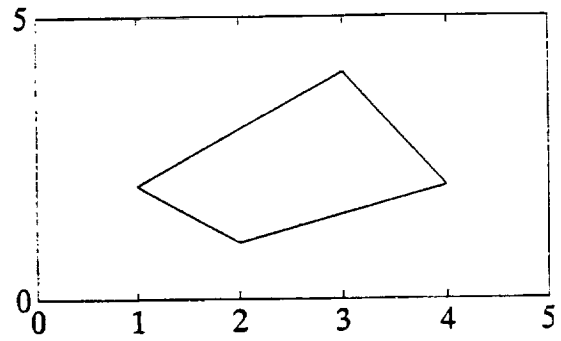
(a)



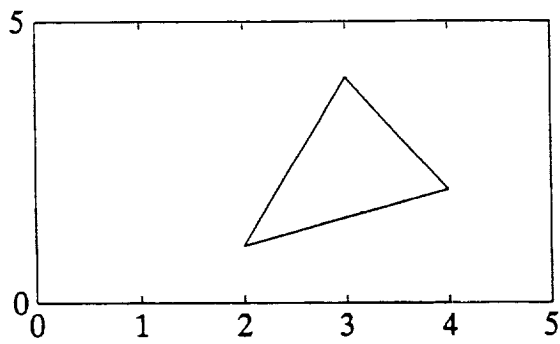
(b)



(c)



(d)



(e)

Figure 3.8 Five possible elements formed by data points listed in Table 3.5

element 1 [2345]		
initial guess	root	damage
0 , 0	diverge	—
1 , 1 *	diverge	—
1 , -1	diverge	—
-1 , -1	diverge	—
-1 , 1	diverge	—

(a)

element 2 [1345]		
initial guess	root	damage
0 , 0	3.0 , 1.0	3.72E+01
1 , 1	3.0 , 1.0	3.72E+01
1 , -1 *	3.0 , 1.0	3.72E+01
-1 , -1	3.0 , 1.0	3.72E+01
-1 , 1	3.0 , 1.0	3.72E+01

(b)

element 3 [1245]		
initial guess	root	damage
0 , 0	1.37 , -0.26	4.62E+01
1 , 1	1.37 , -0.26	4.62E+01
1 , -1 *	1.37 , -0.26	4.62E+01
-1 , -1	1.37 , -0.26	4.62E+01
-1 , 1	1.37 , -0.26	4.62E+01

(c)

element 4 [1235]		
initial guess	root	damage
0 , 0	1.44 , -0.14	4.66E+01
1 , 1	1.44 , -0.14	4.66E+01
1 , -1 *	1.44 , -0.14	4.66E+01
-1 , -1	-1 , 3	—
-1 , 1	-1 , 3	—

(d)

element 5 [1234]		
initial guess	root	damage
0 , 0	1.49 , -0.29	4.66E+01
1 , 1	1.49 , -0.29	4.66E+01
1 , -1 *	1.49 , -0.29	4.66E+01
-1 , -1	1.49 , -0.29	4.66E+01
-1 , 1	diverge	—

(e)

Table 3-6 Comparisons of the predicted damages in example 2, based on different elements and initial guesses (exact damage is 55)

to check the irregularities of the elements and the uniqueness of the mapping just as we treated the cases in example 1. We take the 4 corners of each element in the natural coordinate plane as initial guesses to solve the nonlinear systems, and then determine the associated damages (see Table 3.6 a-e). From these 5 tables, we find that both element 2 and element 3 with regular shapes, where the roots are uniquely determined, can be used to obtain a prediction of the exact damage of 55. An approximation of D can also be obtained from element 4 and element 5, though there are one or two corners that ruin the uniqueness of the the roots due to their geometric distortion. Like example 1, we find that the node (4,3) is closer to the regular corners of the elements 2 and 3 such that the uniqueness of the roots is still valid around these corners. Conversely, node (4,3) is closer to the irregular corner of the element 1, where 3 nodes are collinear and the root cannot converge. The corner that is closest to (4,3) with reference to natural coordinate plane is marked with '*' in each of the Table 3.6 a-e. A similar observation to that which was made considering example 1 can be made here. If the root obtained by the initial guess of the corner which is closest to the predicted node in the natural coordinate plane is the same as that by the guess of (0,0), then the uniqueness of mapping is assumed satisfied and the approximation can be thought reasonable. Thus, except element 1, we have 4 possible approximations of D in this example. Because the prediction node should be as near the element we use as possible for applying linear interpolation functions, the prediction associated with

the root closest to the origin of the natural coordinate plane will be considered the most accurate. As shown in table 3.7, the distance associated with the 3rd element is the shortest and is marked with '*', so the final approximation is determined on the basis of this element.

3.3 Building a Criterion for Prediction

Based on the 2 examples considered in this section, it is not difficult to extend the same approach to problems of higher DOF. Therefore, we can make a criterion for prediction by generalizing the

element	distance to 0
2345	—
1345	3.162
1245	1.396 *
1235	1.451
1234	1.514

Table 3.7 Comparison of the distances from each natural coordinate solved from the nonlinear solver to the origin

observations made as follows :

An ideal isoparametric approximation model should be based on a least distorted element which contains the point to be predicted. However, it is not always possible to use undistorted elements in practical applications, especially for the cases of higher DOF. Accordingly, a unique isoparametric mapping over the whole element often cannot be

obtained. For this reason, only the uniqueness of the mapping over the relatively undistorted part of an element is used by comparing the roots from the nonlinear system when the initial guess of (0,0) is made and when the initial guess of the corner closest to the predicted node with reference to the natural coordinate are used. Thus, even for distorted elements, the "partial uniqueness" of the mapping can result in a reasonable function prediction as long as the predicted node either inside or outside the element is near an undistorted part of the element. When the node to be predicted is outside all possible elements, the final approximation will be determined by the root closest to the origin of the natural coordinate, if several possible approximations are available.

CHAPTER FOUR
NUMERICAL TESTING AND RESULTS

4.1 Sources of Data and the Testing Procedure

This chapter will be devoted to testing the interpolation — extrapolation model using two sets of actual experimental data. The first set of data that is given Table 4.1 was collected from experiments on debris impact on the simulated bumper of the Space Station performed at Marshall Space Flight Center [22]. This data will be used for testing 3-DOF and 4-DOF interpolation models in the following sections. As shown in the Table 4.1, the independent parameters of t, d, θ , and v represent the thickness of the bumper, the diameter of the debris particle, the impact angle, and impact velocity, respectively. These parameters were illustrated in Figure 2.3. Based on these parameters, the dependent variables of the major diameter of the bumper hole (D_{maj}) and the minor diameter of the bumper hole (D_{min}) will be predicted.

In addition, for evaluating the flexibility and the versatility of our model, a second set of data from the field of geology [23] will be used for testing the 5-DOF case, Table 4.2. The data in Table 4.2 are related to the problem of determining the basin magnitude (Y), which essentially is a count of the number of sources in the basin, by

Table 4.1 : Listing of the experimental data of debris
 impact on the simulated bumper of space station

t	d	θ	v	D _{maj}	D _{min}
0.063	0.313	45	6.39	0.81	0.62
0.04	0.313	45	6.51	0.76	0.54
0.04	0.25	45	5.51	0.53	0.43
0.04	0.25	45	7.21	0.53	0.25
0.04	0.3	65	6.45	0.94	0.53
0.04	0.3	65	3.67	0.87	0.47
0.04	0.35	65	3.04	0.98	0.49
0.063	0.25	45	4.11	0.61	0.48
0.063	0.25	45	4.59	0.65	0.49
0.063	0.25	45	5.30	0.6	0.5
0.063	0.3	65	5.85	0.88	0.59
0.063	0.3	65	7.08	1.02	0.64
0.063	0.25	65	6.4	0.87	0.52
0.063	0.25	65	7.4	0.77	0.57
0.063	0.35	65	5.7	1.13	0.66
0.063	0.35	65	6.8	1.4	0.68
0.063	0.35	45	5.85	0.86	0.68
0.04	0.35	45	6.4	0.8	0.6
0.04	0.35	45	6.76	0.84	0.59
0.04	0.187	45	6.36	0.48	0.4
0.04	0.187	45	5.89	0.54	0.42
0.04	0.187	45	4.57	0.45	0.36
0.032	0.25	45	5.52	0.6	0.46
0.032	0.25	45	7.12	0.6	0.43
0.063	0.187	45	4.41	0.48	0.39
0.063	0.187	45	3.23	0.47	0.36
0.063	0.25	45	2.95	0.54	0.43
0.063	0.313	45	4.59	0.73	0.58
0.063	0.313	45	4.34	0.71	0.57
0.04	0.187	45	2.88	0.4	0.3
0.125	0.187	45	3.61	0.4	0.31
0.04	0.25	45	4.47	0.53	0.42
0.04	0.313	45	7.0	0.77	0.52
0.04	0.313	45	6.98	0.84	0.57
0.04	0.375	45	6.49	0.83	0.69

Table 4.2 : Listing of geological observed data
used to determine basin magnitude

elevation	relief	area	length of the stream	drainage density	basin magnitude
720	570	7	154	2200	14
670	610	3	80	2667	6
860	550	11	84	763	5
870	610	11	122	1110	7
730	570	14	185	1321	11
690	590	12	200	1667	14
880	640	11	170	1545	12
760	690	28	340	1215	18
820	600	5	100	2000	6
720	480	3	80	2667	5
670	670	19	290	1526	17
660	600	5	90	1800	5
830	660	18	260	1444	22
780	620	17	111	652	7
750	740	15	184	1227	15
770	630	21	227	1080	17
750	570	4	60	1500	5
750	580	20	259	1295	18
740	760	9	62	689	14
750	740	6	95	1583	21
750	760	11	105	954	22
740	770	32	350	1094	23
940	510	21	232	1105	28
700	600	23	266	1156	42
810	580	44	390	886	22
920	500	13	142	1092	10
920	490	12	145	1208	11
790	605	33	253	766	12
860	550	23	241	1048	13
860	630	87	702	807	31
880	520	37	288	778	18
780	460	17	162	953	13
720	440	8	67	838	4
780	300	3	52	1733	5
700	460	10	121	1210	9
680	520	26	220	846	13
820	520	8	123	1537	10

(continued)

Table 4.2 (continued)

elevation	relief	area	length of the stream	drainage density	basin magnitude
710	520	24	238	992	13
800	440	19	231	1216	13
700	510	16	178	1113	11
675	570	18	168	933	12
740	510	8	65	812	4
740	520	31	334	1078	17
770	600	21	184	876	9
820	520	11	136	1237	8
850	490	22	233	1059	13
820	629	34	410	1206	22
820	510	11	149	1354	10
680	640	46	348	757	19
660	789	55	382	695	27

considering the 6 following independent variables; the elevation of the basin outlet (x_1), the relief of the basin (x_2), the basin area (x_3), total length of the stream in the basin (x_4), and drainage density (x_5), which is defined as total length of the streams in the basin divided by the basin area.

The interpolation/extrapolation procedure was tested in following manner. The dependent variable (say, diameter or basin magnitude) of each set of data is assumed unknown and then is predicted based on the remaining data. For example, in Table 4-1, suppose the major diameter (or minor diameter) associated with the first set of data is set as unknown, then the rest of 34 sets of data will be used to predict it. Similarly, the major diameters associated with the second, ..., etc. set of the data will be predicted by the others.

For simplicity, each prediction was based on the $2^n + 1$ closest elements (n = number of DOF). As was suggested in chapter 3, the final prediction was based on the element that could produce unique natural coordinate roots after solving the nonlinear mapping equations by two initial guesses. If more than one element produced unique roots in the test, then the final prediction was taken as that produced by the element whose natural coordinate roots were closest to the origin of the natural coordinate system. For cases where no unique mapping was found in all the tested elements, then the roots closest to the origin were used to make the prediction.

Predicted values will now be compared with that measured. A computer program 'ISOMODEL' that implements the algorithms derived in Chapters 2 and 3 was written in Turbo Pascal 5.5 to perform all the

tests on IBM PC 386 machine. This program is listed in the appendix. Because multiple linear regression is a standard and well accepted technique for this sort of problem, the predicted results generated by the statistical software package SAS [24] using the IBM 3090 are compared with the results generated by the isoparametric model. The linear functions that SAS used are of the form :

$$Y = a_0 + a_1 X_1 + a_2 X_2 + \dots + a_n X_n$$

where Y is dependent variable, X_i ($i = 1..n$) are independent variables, and a_i ($i = 0..n$) are coefficients to be determined by the Least Square method.

4.2 Testing for 3-D Case and the Results

In this section, the 3-DOF case is tested by using a subset of Table 4.1, where there are just 26 sets of data that correspond to $\theta = 45$. We predict the major diameter (D_{maj}) first, and then the minor diameter (D_{min}) by the same parameters. The results of prediction on major diameter and minor diameter, compared with those predicted by SAS, are respectively shown in Table 4.3 and 4.4.

As Table 4.3 and Table 4.4 show, the average error of the isoparametric model (9.37%) is greater than that of SAS (5.61%) in absolute value for the prediction of the major diameter. For the prediction of the minor diameter, SAS also has less average error (9.64%) than the isoparametric model (11.68%). The few wild predicted values of the isoparametric model can be attributed to a fatal distortion of elements or possibly to some scatter in the experimental

Table 4.3 : Listing of the prediction of major diameter
by the isoparametric model and SAS vs. the
measured value

No.	MEASURED	PREDICTED BY		PREDICTED BY	
	D _{maj}	ISOMODEL	ERROR%	SAS	ERROR%
1	0.81	0.79	-2.47	0.76	-6.17
2	0.76	0.76	0.00	0.75	-1.32
3	0.53	0.58	11.32	0.60	13.21
4	0.53	0.67	26.42	0.63	18.87
5	0.61	0.62	1.64	0.58	-4.92
6	0.65	0.62	-4.62	0.59	-9.23
7	0.6	0.68	13.33	0.60	0.00
8	0.86	0.86	0.00	0.83	-3.49
9	0.8	0.82	2.50	0.83	3.75
10	0.84	0.86	2.38	0.83	-1.19
11	0.48	0.57	18.75	0.48	0.00
12	0.54	0.47	-12.96	0.47	-12.96
13	0.45	0.48	-2.22	0.45	0.00
14	0.6	0.53	-11.67	0.60	0.00
15	0.6	0.48	-20.00	0.62	3.33
16	0.48	0.56	16.67	0.45	-6.25
17	0.47	0.46	-2.13	0.43	-8.51
18	0.54	0.51	-5.56	0.56	3.70
19	0.73	0.72	0.00	0.72	-1.37
20	0.71	0.72	1.41	0.72	1.41
21	0.4	0.37	-7.50	0.42	5.00
22	0.4	0.47	17.50	0.46	15.00
23	0.53	0.29	-45.28	0.58	9.43
24	0.77	0.84	9.09	0.76	-1.30
25	0.84	0.77	-8.33	0.76	-9.52
26	0.83	0.88	0.00	0.88	6.02
			Ave= 9.37%	Ave=5.61%	

Table 4.4 : Listing of the prediction of minor diameter
by the isoparametric model and SAS vs. the
measured value

No.	MEASURED D _{ma} J	PREDICTED ISOMODEL	ERROR%	PREDICTED BY SAS	ERROR%
1	0.62	0.64	3.23	0.56	-9.68
2	0.54	0.56	3.70	0.55	1.85
3	0.43	0.45	4.65	0.45	4.65
4	0.25	0.51	104.00	0.44	76.00
5	0.48	0.47	-2.08	0.46	-4.17
6	0.49	0.49	0.00	0.46	-6.12
7	0.50	0.51	2.00	0.46	-8.00
8	0.68	0.66	-2.94	0.63	-7.35
9	0.60	0.65	8.33	0.62	3.33
10	0.59	0.62	5.08	0.62	5.08
11	0.40	0.44	10.00	0.33	-17.50
12	0.42	0.39	-7.14	0.34	-19.05
13	0.36	0.37	2.78	0.34	-5.56
14	0.46	0.43	-6.52	0.44	-4.35
15	0.43	0.16	-62.79	0.43	0.00
16	0.39	0.40	2.56	0.35	-10.26
17	0.36	0.35	-2.78	0.36	0.00
18	0.43	0.46	6.98	0.47	9.30
19	0.58	0.58	0.00	0.57	-1.72
20	0.57	0.57	0.00	0.57	0.00
21	0.30	0.30	0.00	0.35	16.67
22	0.31	0.36	16.13	0.37	19.35
23	0.42	0.33	-21.43	0.45	7.14
24	0.52	0.57	9.62	0.55	5.77
25	0.57	0.52	-8.77	0.55	-3.51
26	0.69	0.62	-10.14	0.66	-4.35
			Ave=11.68%		Ave=9.64%

data. The comparisons of the results are shown in Figure 4.1 and Figure 4.2.

4.3 Testing for 4-D Case and the Results

For the 4-DOF case, we take all the 4 independent parameters and all the 35 sets of data in Table 4.1 into consideration to test the isoparametric model. The results are listed in Table 4.5 and Table 4.6. By comparing the average errors, the isoparametric model has less accurate predictions than SAS in major diameter but more accurate than SAS in minor diameter. Figure 4.3 and 4.4 also shows that the predictions made by SAS are more uniformly scattered along the 45 degree line than those by the isoparametric model. The few wild predicted values of the isoparametric model could be caused by serious distortion of elements as well as the numerical errors from the nonlinear system solver due to the higher order nonlinear terms when Newton's method applied. The inconsistency of part of the experimental data could also exaggerate the deviations for both models.

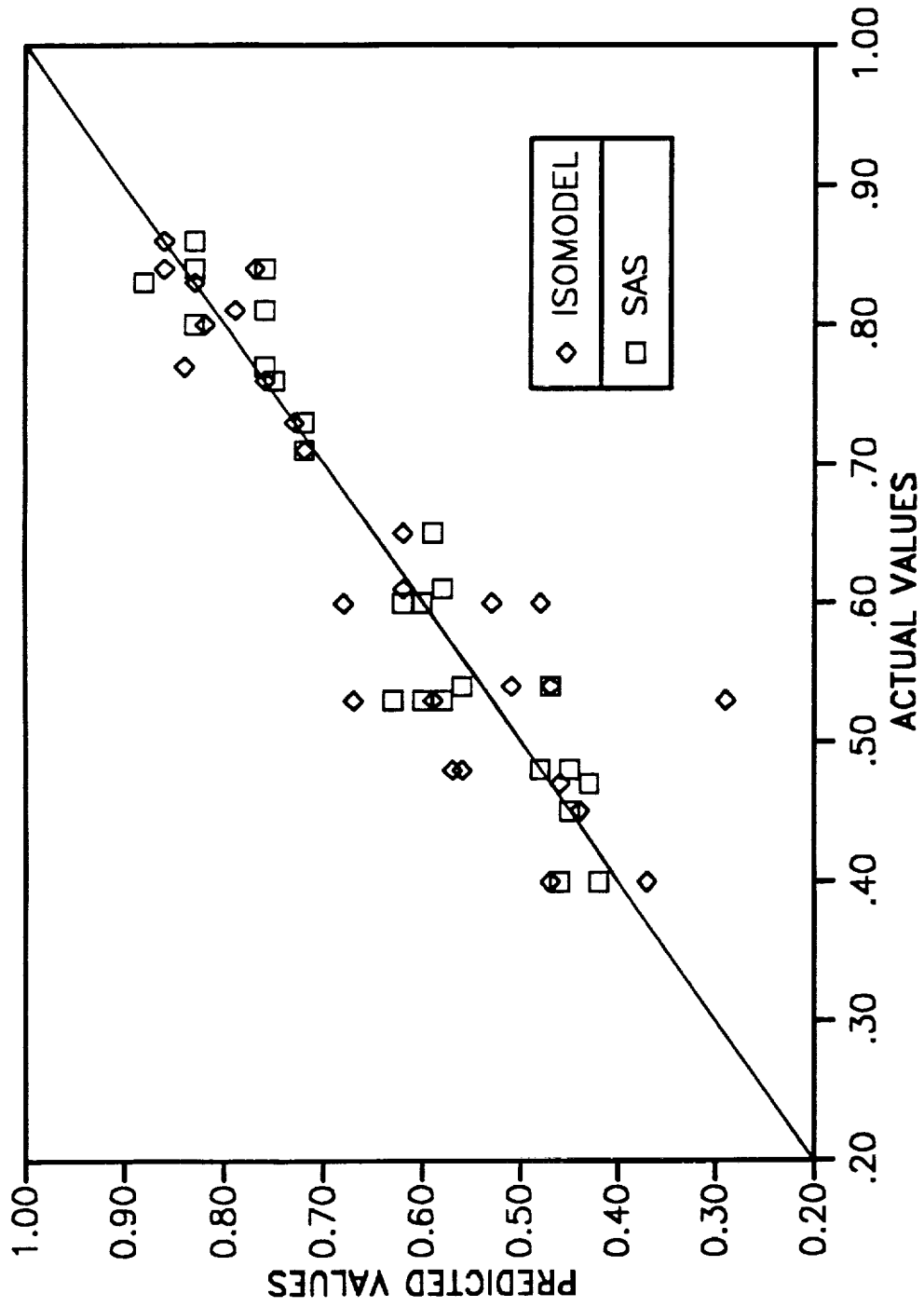


Figure 4.1 Predictions by the isoparametric model and SAS versus observed major diameter for the 3-DOF test. Diagonal line indicates predicted equal observed.

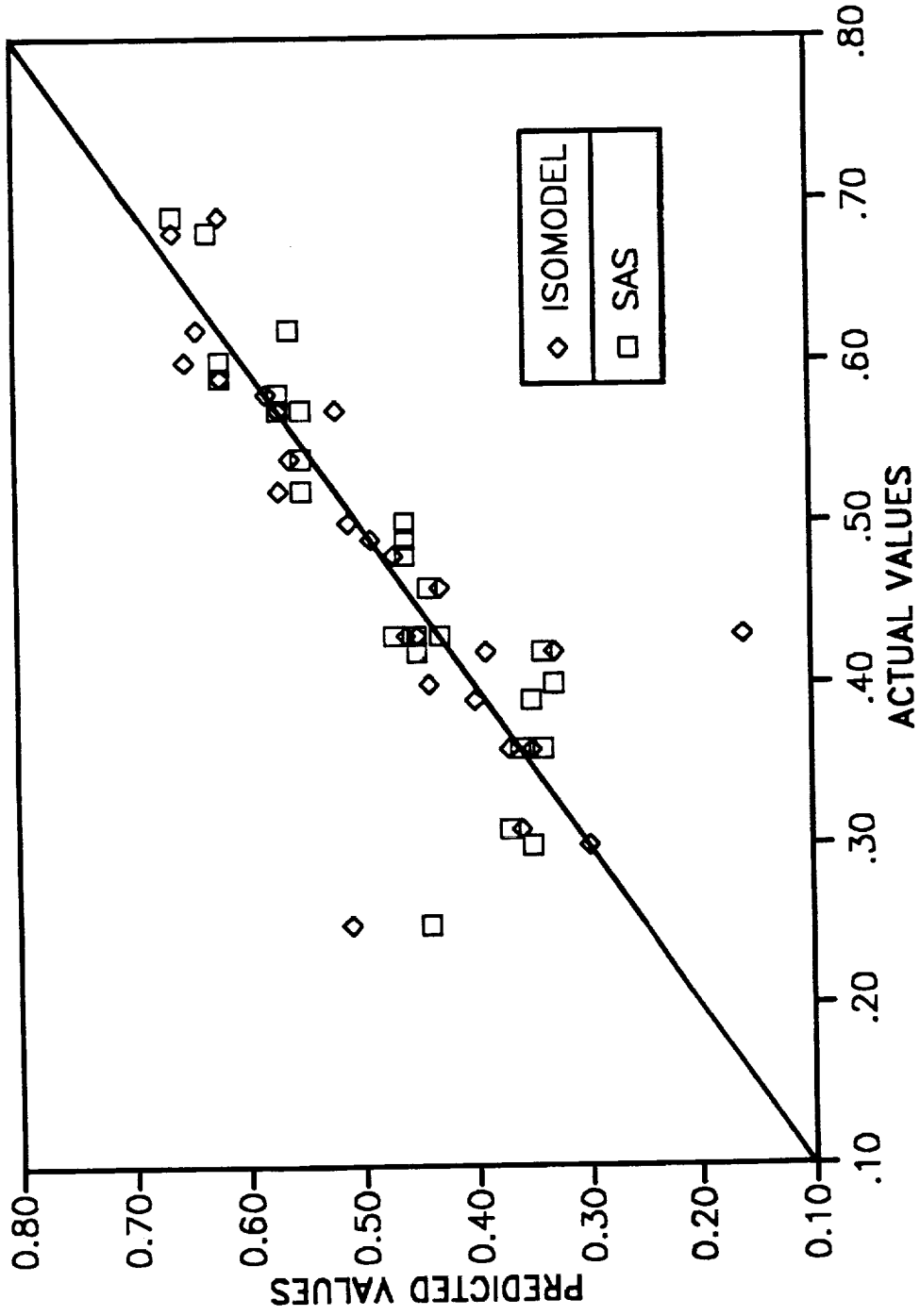


Figure 4.2 Predictions by the isoparametric model and SAS versus observed minor diameter for the 3-DOF test. Diagonal line indicates predicted equal observed.

Table 4.5 : Listing of the prediction of major diameter
by the isoparametric model and SAS vs. the
measured value

No.	MEASURED	PREDICTED	ERROR%	PREDICTED BY	ERROR%
	D _{maj}	ISOMODEL		SAS	
1	0.81	0.80	-1.23	0.78	-3.70
2	0.76	0.73	-3.95	0.76	0.00
3	0.53	0.59	11.32	0.59	11.32
4	0.53	0.75	41.51	0.63	18.87
5	0.94	0.76	-19.15	0.97	3.19
6	0.87	0.90	3.45	0.91	4.60
7	0.98	0.95	-3.06	1.01	3.06
8	0.61	0.62	1.64	0.58	-4.92
9	0.65	0.61	-6.15	0.59	-9.23
10	0.60	0.70	16.67	0.61	1.67
11	0.88	0.98	11.36	0.98	11.36
12	1.02	1.08	5.88	1.01	-0.98
13	0.87	0.74	-14.94	0.87	0.00
14	0.77	1.17	51.95	0.89	15.58
15	1.13	0.94	-16.81	1.09	-3.54
16	1.40	1.05	-25.00	1.12	-20.00
17	0.86	0.99	15.12	0.85	-1.16
18	0.80	0.73	-8.75	0.84	5.00
19	0.84	0.82	-2.38	0.85	1.19
20	0.48	0.54	12.50	0.46	-4.17
21	0.54	0.49	-9.26	0.45	-16.67
22	0.45	0.48	6.67	0.42	-6.67
23	0.60	0.58	-3.33	0.58	-3.33
24	0.60	0.76	26.67	0.62	3.33
25	0.48	0.63	31.25	0.44	-8.33
26	0.47	0.48	2.13	0.41	-12.77
27	0.54	0.61	12.96	0.55	1.85
28	0.73	0.73	0.00	0.74	1.37
29	0.71	0.71	0.00	0.73	2.82
30	0.4	0.45	12.50	0.38	-5.00
31	0.4	0.77	92.50	0.48	20.00
32	0.53	0.61	15.09	0.57	7.55
33	0.77	0.84	9.09	0.77	0.00
34	0.84	0.77	-8.33	0.77	-8.33
35	0.83	0.83	0.00	0.90	8.43
			Ave=14.36%	Ave=6.57%	

Table 4.6 : Listing of the prediction of minor diameter
by the isoparametric model and SAS vs. the
measured value

No.	MEASURED D _{ma j}	PREDICTED ISOMODEL	ERROR%	PREDICTED BY SAS	ERROR%
1	0.62	0.66	6.45	0.58	-6.45
2	0.54	0.55	1.85	0.56	3.70
3	0.43	0.45	4.65	0.44	2.33
4	0.25	0.48	92.00	0.47	88.00
5	0.53	0.54	1.89	0.56	5.66
6	0.47	0.46	-2.13	0.51	8.51
7	0.49	0.52	6.12	0.57	16.33
8	0.48	0.47	-2.08	0.45	-6.25
9	0.49	0.49	0.00	0.46	-6.12
10	0.50	0.52	4.00	0.47	-6.00
11	0.59	0.59	0.00	0.57	-3.39
12	0.64	0.62	-3.13	0.60	-6.25
13	0.52	0.55	5.77	0.51	-1.92
14	0.57	0.54	-5.26	0.53	-7.02
15	0.66	0.65	-1.52	0.65	-1.52
16	0.68	0.70	2.94	0.67	-1.47
17	0.68	0.66	-2.94	0.63	-7.35
18	0.60	0.61	1.67	0.61	1.67
19	0.59	0.60	1.70	0.62	5.08
20	0.40	0.42	5.00	0.36	-10.00
21	0.42	0.41	-2.38	0.36	-14.29
22	0.36	0.37	2.78	0.33	-8.33
23	0.46	0.46	0.00	0.43	-6.52
24	0.43	0.47	9.30	0.46	6.98
25	0.39	0.40	2.56	0.36	-7.69
26	0.36	0.39	8.33	0.34	-5.56
27	0.43	0.48	11.63	0.43	0.00
28	0.58	0.58	0.00	0.55	-5.17
29	0.57	0.57	0.00	0.55	-3.51
30	0.3	0.36	20.00	0.30	0.00
31	0.31	0.46	48.39	0.43	38.71
32	0.42	0.40	-4.76	0.43	2.38
33	0.52	0.57	9.62	0.56	7.69
34	0.57	0.52	-8.77	0.56	-1.75
35	0.69	0.64	-7.25	0.65	-5.80
			Ave=8.20%	Ave=8.84%	

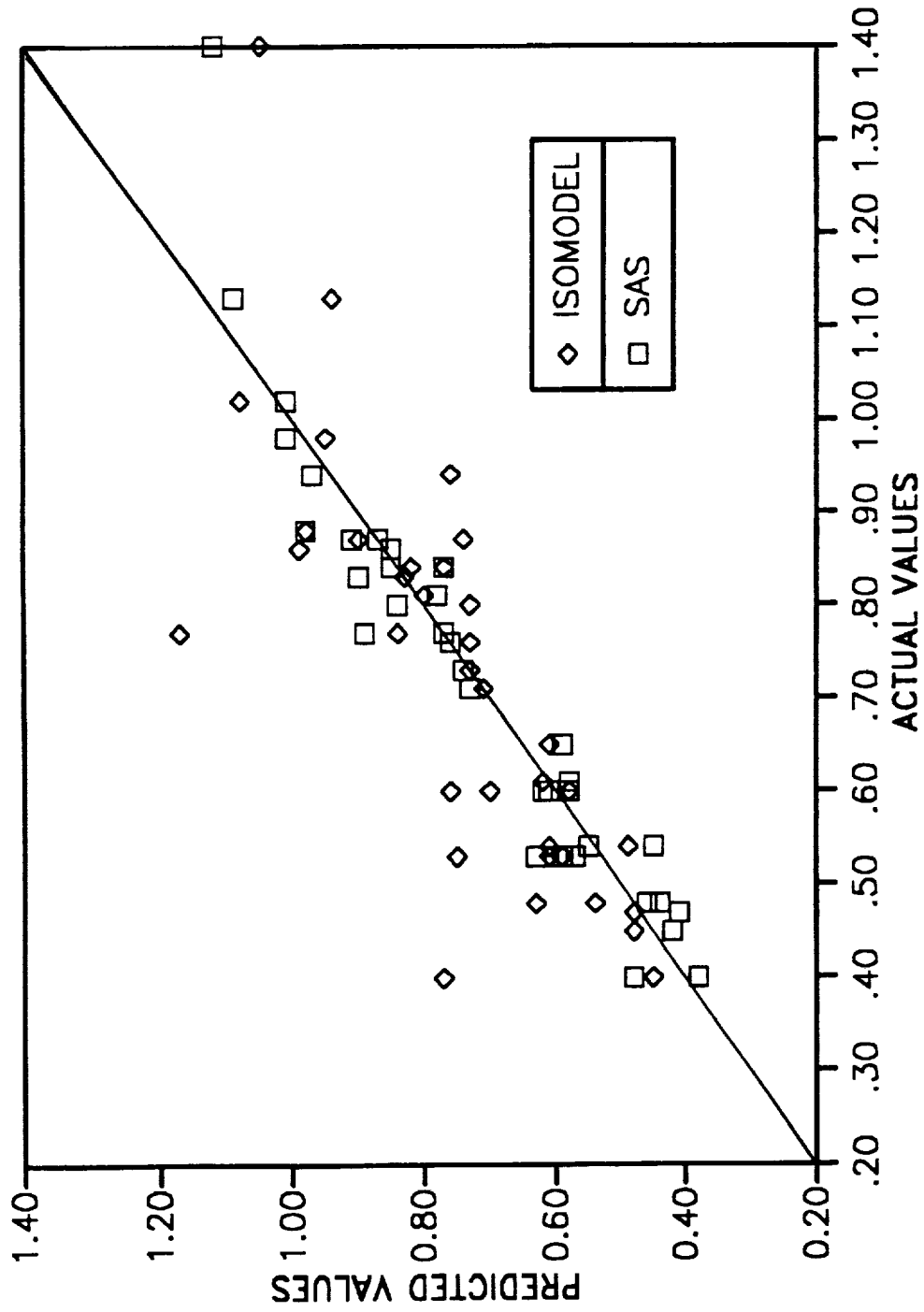


Figure 4.3 Predictions by the isoparametric model and SAS versus observed major diameter for the 4-DOF test. Diagonal line indicates predicted equal observed.

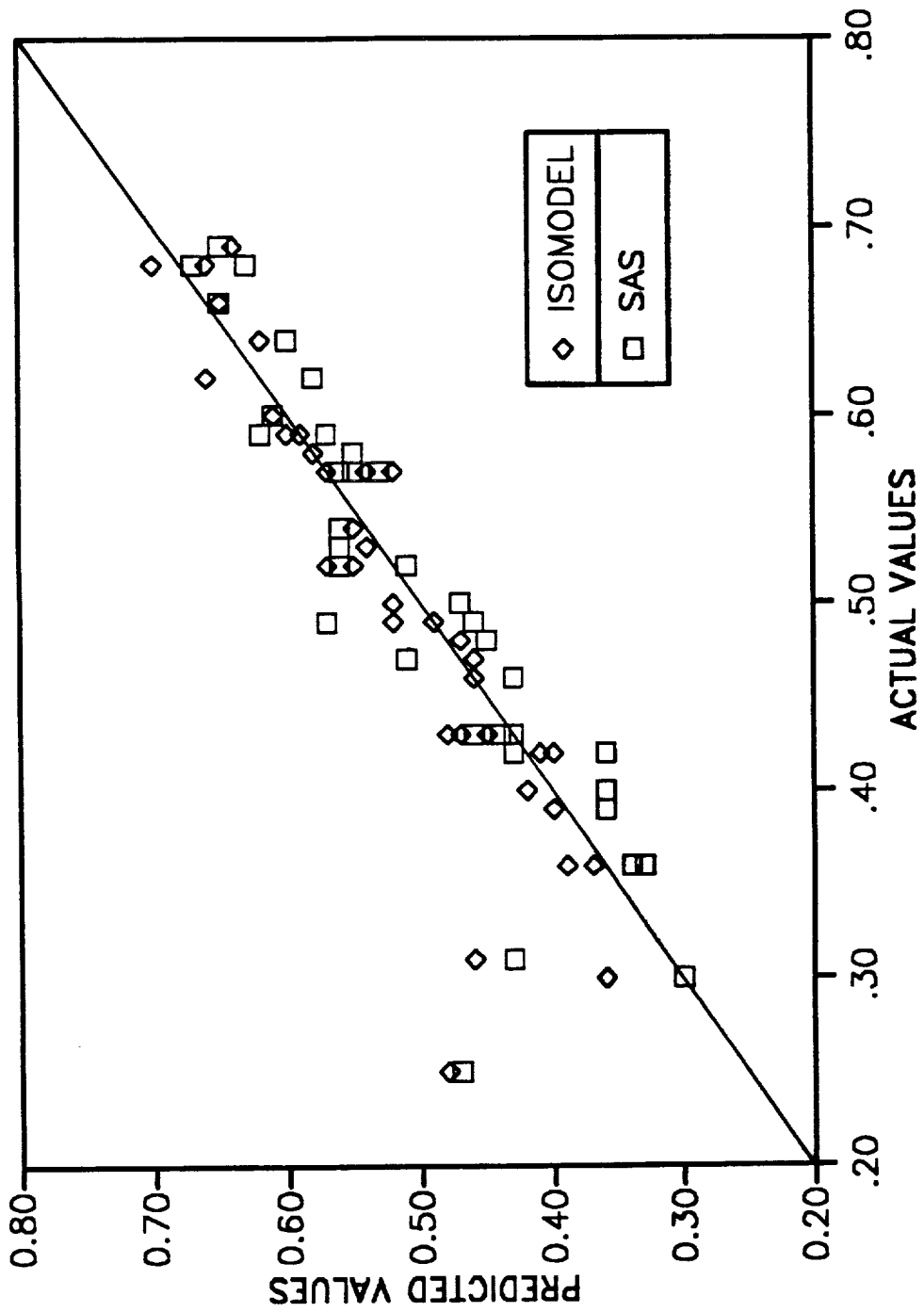


Figure 4.4 Predictions by the isoparametric model and SAS versus observed minor diameter for the 4-DOF test. Diagonal line indicates predicted equal observed.

4.4 Testing for 5-D Case

In this section, the 50 sets of 5-DOF data of Table 4.2 were used to test the isoparametric model. The results are listed in Table 4.7. The isoparametric model and SAS have much higher average errors (48.20% and 25.48%, respectively) than in the previous cases. Referring to Figure 4.5, it is can be seen that SAS can make a closer prediction to the observed values than the isoparametric model. More wild predicted values by the isoparametric model appear than those in 3-DOF and 4-DOF cases. This could be partially due to the effects of distortion on elements with higher DOF, and partially due to the experimental scatter.

Table 4.7 : Listing of the prediction of basin magnitude
by the isoparametric model and SAS vs. the
measured value

No.	MEASURED D _{maj}	PREDICTED ISOMODEL	ERROR%	PREDICTED BY SAS	ERROR%
1	14	10.17	-27.36	11.53	-17.64
2	6	4.04	-32.67	7.43	23.83
3	5	5.27	5.40	8.69	73.80
4	7	-2.15	-130.71	11.76	68.00
5	11	11.53	4.82	13.81	25.55
6	14	11.19	-20.07	14.59	6.79
7	12	15.81	31.75	14.65	22.08
8	18	17.30	-3.89	23.57	30.94
9	6	4.22	-29.67	9.57	59.50
10	5	2.03	-59.40	4.63	-7.40
11	17	11.27	-33.71	21.26	25.06
12	5	6.52	25.00	9.26	85.20
13	22	17.34	-21.18	19.56	-11.09
14	7	-6.4	-191.43	10.76	53.71
15	15	36.67	144.47	17.42	16.13
16	17	23.50	38.24	16.76	-1.41
17	5	7.92	58.40	7.56	51.20
18	18	26.86	49.22	17.56	-2.44
19	14	20.58	47.00	12.37	-11.64
20	21	12.60	-40.00	12.91	-38.52
21	22	14.29	-35.05	14.16	-35.64
22	23	22.54	-2.00	25.27	9.87
23	28	12.78	-54.36	14.52	-48.14
24	42	15.94	-62.05	18.01	-57.12
25	22	19.25	-12.50	21.41	-2.68
26	10	10.02	0.20	10.28	2.80
27	11	8.29	-24.64	10.25	-6.81
28	12	0.90	-92.50	15.77	31.42
29	13	13.53	4.08	15.59	19.92
30	31	51.63	66.55	33.06	6.65
31	18	5.34	-70.33	15.30	-15.00
32	13	10.73	-17.46	10.00	-23.08
33	4	10.44	161.00	5.65	41.25
34	5	-1.3	-126.00	0.91	-81.80
35	9	4.0	-55.56	8.37	-7.00

(continued)

(Table 4.7 continued)

No.	MEASURED	PREDICTED	ERROR%	PREDICTED BY	ERROR%
	Dmaj	ISOMODEL		SAS	
36	13	-3.8	-129.23	13.18	1.38
37	10	7.02	-29.80	9.64	-3.60
38	13	14.01	7.77	14.55	11.92
39	13	18.77	44.38	13.10	0.77
40	11	9.56	-13.09	12.00	9.09
41	12	12.43	3.58	12.54	4.50
42	4	6.23	55.75	7.10	77.50
43	17	23.93	40.76	19.05	12.06
44	9	10.04	11.56	13.71	52.33
45	8	10.72	34.00	10.43	30.38
46	13	29.29	125.31	13.97	7.46
47	22	0.63	-97.13	25.50	15.91
48	10	8.80	-12.00	10.82	8.20
49	19	21.12	11.16	19.67	3.53
50	27	22.76	-15.70	23.13	-14.33

Ave=48.20% Ave=25.48%

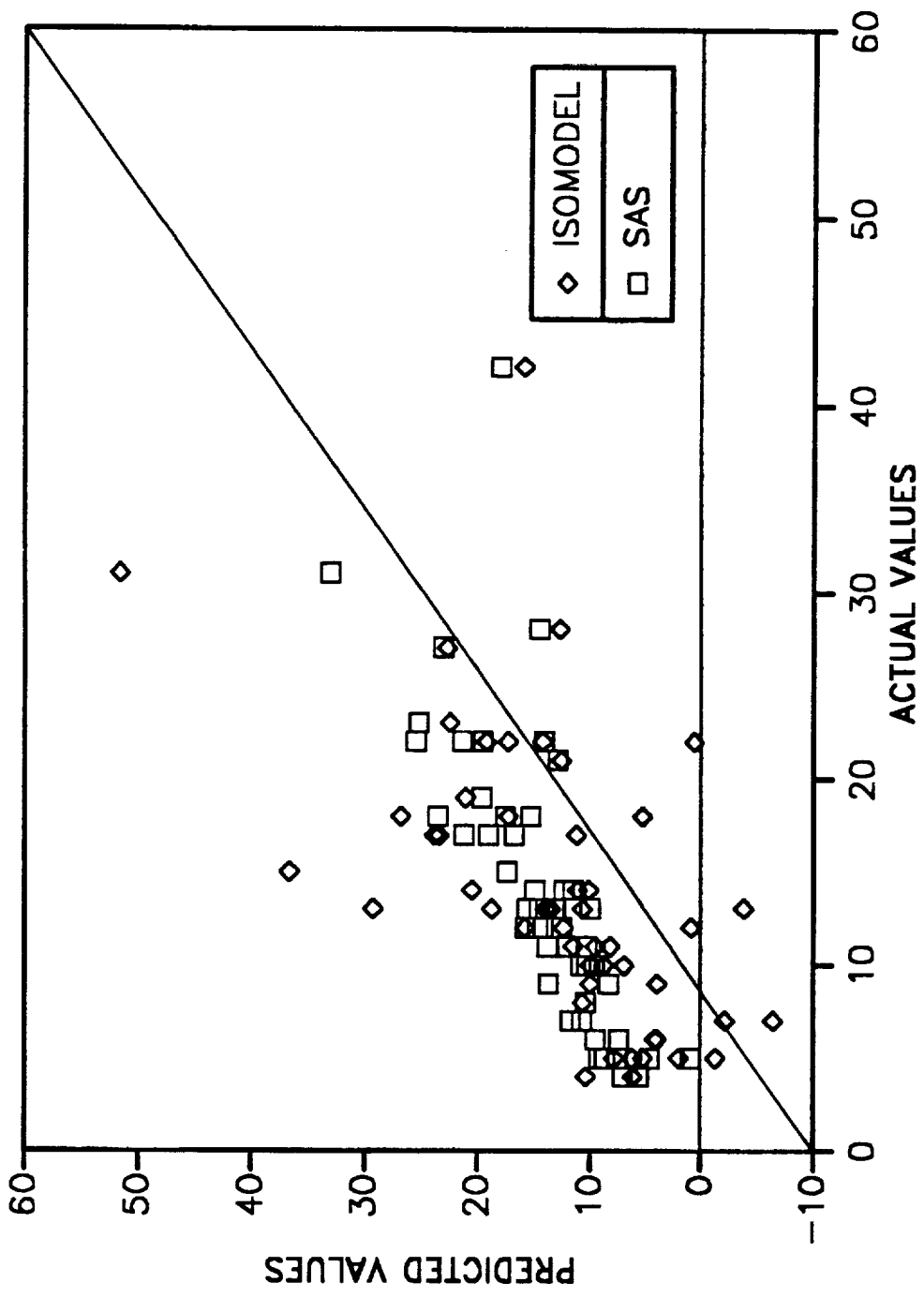


Figure 4.5 Predictions by the isoparametric model and SAS versus observed basin number for the 5-DOF test. Diagonal line indicates predicted equal observed.

CHAPTER FIVE
CONCLUSIONS AND RECOMMENDATIONS

This thesis has proposed a unique technique for multivariable analysis based on extending isoparametric concept of FEM. By extending the concept of shape functions defined by geometric coordinates in 2-D and 3-D space to that of those defined by physical coordinates in n-D space, a multivariable interpolation model was developed to analyze scattered data with arbitrary DOF. A Pascal program called ISOMODEL, based on the methodology derived in Chapter 3, was tested for the 3-D, 4-D and 5-D cases by two sets of actual experimental data in Chapter 4. Compared with the results from the statistical software SAS, the isoparametric model was less accurate in the 3-D and 4-D cases, but the difference of the average errors of these two models is not much. However, in the cases of higher DOF, the isoparametric model is far less accurate than SAS. Geometrically, this is because the isoparametric model is too sensitive to the distortion of the elements. Technically, the isoparametric model involves solving nonlinear systems of equations, which often causes remarkable numerical errors in treating higher order nonlinear terms. At the same time, errors originating from the scatter in experimental observations and measurements could also increase the prediction errors.

It should be noticed that the points to be predicted were excluded and then estimated by the remaining sets of data when the isoparametric model was tested. On the contrary, the points to be predicted, together with the remaining data points, were used to determine the linear regression model when SAS was tested. This could be another factor to cause some bias in predictions between the isoparametric model and SAS.

In order to improve the weak points of the isoparametric model, further study is required. It is recommended that an alternate way be developed for finding the natural coordinates in the isoparametric mapping such that the errors of solving nonlinear systems can be reduced. Finding a more effective way to find the least distorted elements from the nodal points could help lower the possibility of the wild predictions.

APPENDIX 2 A LISTING OF THE COMPUTER CODE. (LISTING NOT PROVIDED IF FLOPPY DISKS INCLUDED)

MLIBLAST.BAS

```
,
,
'  MLIBLAST.BAS
'  Source code for the main program that runs the other programs
,
,
CLS
COLOR 15, 0
LOCATE 12, 37
PRINT "MLIBLAST"
seed% = ((TIMER * 65536) / 86400) - 32768
RANDOMIZE seed%
FOR I% = 1 TO 2000
testcolor# = RND
IF testcolor# >= .666 THEN colornum% = 15
IF testcolor# <= .333 THEN colornum% = 9
IF testcolor# > .333 AND testcolor# < .666 THEN colornum% = 12
COLOR colornum%, 0
row% = 1! + RND * 22!
col% = 1! + RND * 79!
  IF row% < 11 OR row% > 13 THEN
    LOCATE row%, col%
    PRINT CHR$(219)
  ELSE
    IF col% < 36 OR col% > 45 THEN
      LOCATE row%, col%
      PRINT CHR$(219)
    END IF
  END IF
NEXT I%
MainMenu:
COLOR 15, 9
CLS
LOCATE 1, 3
PRINT "Please Enter The ";
COLOR 12, 9
PRINT "Number ";
COLOR 15, 9
PRINT "Associated With the Desired Action:"
LOCATE 5, 3
COLOR 12, 9
PRINT "1. ";
COLOR 15, 9
PRINT "Add Data To The Database"
LOCATE 7, 3
COLOR 12, 9
PRINT "2. ";
COLOR 15, 9
PRINT "Predict MLI Damage"
LOCATE 9, 3
COLOR 12, 9
PRINT "3. ";
COLOR 15, 9
PRINT "Quit"
LOCATE 13, 3
COLOR 12, 9
INPUT Choice%
IF Choice% < 1 OR Choice% > 3 THEN GOTO MainMenu
,
IF Choice% = 1 THEN
```

MLIBLAST.BAS

```
SHELL " database.exe"
GOTO MainMenu
END IF
'
IF Choice% = 2 THEN
CLS
COLOR 15, 9
LOCATE 1, 3
PRINT "Please Enter The ";
COLOR 12, 9
PRINT "Number ";
COLOR 15, 9
PRINT "Associated With the Desired Prediction Scheme:"
LOCATE 5, 3
COLOR 12, 9
PRINT "1. ";
COLOR 15, 9
PRINT "1/R^N Interpolation Scheme"
LOCATE 7, 3
COLOR 12, 9
PRINT "2. ";
COLOR 15, 9
PRINT "Polynomial Interpolation Scheme"
LOCATE 9, 3
COLOR 12, 9
PRINT "3. ";
COLOR 15, 9
PRINT "Nondimensional Parameter Scheme"
LOCATE 11, 3
COLOR 12, 9
PRINT "4. ";
COLOR 15, 9
PRINT "Return to the Main Menu"
LOCATE 13, 3
COLOR 12, 9
INPUT Choice%
IF Choice% = 1 THEN SHELL "invrmeth.exe"
IF Choice% = 2 THEN SHELL "polymeth.exe"
IF Choice% = 3 THEN SHELL "nondimen.exe"
IF Choice% = 4 THEN GOTO MainMenu
GOTO MainMenu
END IF
IF Choice% = 3 THEN
COLOR 15, 0
CLS
END
END IF
```

DATABASE.BAS

```
,
,
,   DATABASE.BAS
,   Source code for the data base program
,
,
ON KEY(10) GOSUB DataInputProblem
AddData:
COLOR 11, 0
CLS
LOCATE 1, 3
PRINT "Enter MLI Test Data File Name: ";
COLOR 12, 0
INPUT "", MLITestDataFile$
OPEN MLITestDataFile$ FOR APPEND AS #1
AddData1:
KEY(10) ON
GOSUB DisplayTemplate
,
'prompt user for data
,
COLOR 12, 0
LOCATE 25, 1
PRINT "Please Enter Data At The Cursor - Press F10 and ENTER To Restart Input";
COLOR 12, 0
,
LOCATE 2, 10
INPUT "", TestID$
,
LOCATE 2, 37
INPUT "", DataSource$
,
,
LOCATE 2, 62
INPUT "", TestDate$
,
,
LOCATE 5, 18
INPUT "", BumperMaterial$
,
,
LOCATE 5, 48
INPUT "", BumperThickness#
,
,
LOCATE 5, 74
INPUT "", BumperStandOff#
,
,
LOCATE 8, 25
INPUT "", PressureWallMaterial$
,
,
LOCATE 8, 74
INPUT "", PressureWallThickness#
,
,
LOCATE 11, 22
INPUT "", ProjectileMaterial$
```

DATABASE.BAS

```
,
LOCATE 11, 74
INPUT "", ProjectileDiameter#
,
,
LOCATE 14, 15
INPUT "", ImpactAngle#
,
,
LOCATE 14, 70
INPUT "", ProjectileVelocity#
,
,
LOCATE 17, 33
INPUT "", BumperMajorAxis#
,
,
LOCATE 17, 70
INPUT "", BumperMinorAxis#
,
,
LOCATE 20, 20
INPUT "", MLIHoleDiam#
,
,
LOCATE 20, 70
INPUT "", MLIMassLoss#
,
,
LOCATE 23, 39
INPUT "", PressWallMajAxis#
,
,
LOCATE 23, 70
INPUT "", PressWallMinAxis#
,
,
KEY(10) OFF
LOCATE 25, 1
PRINT "
LOCATE 25, 1
COLOR 9, 0
PRINT "OK to write this data to the database (y/n)? ";
COLOR 12, 0
INPUT "", answer$
COLOR 15, 0
answer$ = LCASE$(answer$)
IF answer$ = "y" THEN
  GOSUB WriteDataToFile
  CLS
  LOCATE 25, 1
  COLOR 9, 0
  PRINT "Do you wish to enter more data at this time (y/n)? ";
  COLOR 12, 0
  INPUT "", answer$
  answer$ = LCASE$(answer$)
  COLOR 15, 0
  IF answer$ = "y" THEN
    GOTO AddData1
  ELSE
```

DATABASE.BAS

```
    GOTO Finish
END IF
ELSE
CLS
LOCATE 25, 1
COLOR 9, 0
PRINT "Do you wish to enter more data at this time (y/n)? ";
COLOR 12, 0
INPUT "", answer$
COLOR 15, 0
IF answer$ = "y" THEN
    GOTO AddData1
ELSE
    GOTO Finish
END IF
END IF
Finish:
END
DisplayTemplate:
CLS
COLOR 11, 0
',
'field 1
',
LOCATE 2, 1
PRINT "Test ID"
BoxRow% = 1
BoxColumn% = 9
BoxLength% = 10
GOSUB BoxDraw
',
'field 2
',
LOCATE 2, 24
PRINT "Data Source"
BoxRow% = 1
BoxColumn% = 36
BoxLength% = 10
GOSUB BoxDraw
',
'field 3
',
LOCATE 2, 51
PRINT "Test Date"
BoxRow% = 1
BoxColumn% = 61
BoxLength% = 18
GOSUB BoxDraw
',
'field 4
',
LOCATE 5, 1
PRINT "Bumper Material"
LOCATE 6, 7: PRINT "Name"
BoxRow% = 4
BoxColumn% = 17
BoxLength% = 10
GOSUB BoxDraw
',
'field 5
```

DATABASE.BAS

```
,  
LOCATE 5, 30  
PRINT "Bumper Thickness"  
LOCATE 6, 36  
PRINT "(in)"  
BoxRow% = 4  
BoxColumn% = 47  
BoxLength% = 6  
GOSUB BoxDraw  
,  
'field 6  
,  
LOCATE 5, 56  
PRINT "Bumper Stand-Off"  
LOCATE 6, 62  
PRINT "(in)"  
BoxRow% = 4  
BoxColumn% = 73  
BoxLength% = 6  
GOSUB BoxDraw  
,  
'field 7  
,  
LOCATE 8, 1  
PRINT "Pressure Wall Material"  
LOCATE 9, 10: PRINT "Name"  
BoxRow% = 7  
BoxColumn% = 24  
BoxLength% = 10  
GOSUB BoxDraw  
,  
'field 8  
,  
LOCATE 8, 48  
PRINT "Pressure Wall Thickness"  
LOCATE 9, 58  
PRINT "(in)"  
BoxRow% = 7  
BoxColumn% = 73  
BoxLength% = 6  
GOSUB BoxDraw  
,  
'field 9  
,  
LOCATE 11, 1  
PRINT "Projectile Material"  
LOCATE 12, 9: PRINT "Name"  
BoxRow% = 10  
BoxColumn% = 21  
BoxLength% = 10  
GOSUB BoxDraw  
,  
'field 10  
,  
LOCATE 11, 53  
PRINT "Projectile Diameter"  
LOCATE 12, 61  
PRINT "(in)"  
BoxRow% = 10  
BoxColumn% = 73
```

DATABASE.BAS

```
BoxLength% = 6
GOSUB BoxDraw
',
'field 10
',
LOCATE 14, 1
PRINT "Impact Angle"
LOCATE 15, 2
PRINT "(degrees)"
BoxRow% = 13
BoxColumn% = 14
BoxLength% = 6
GOSUB BoxDraw
',
'field 11
',
LOCATE 14, 49
PRINT "Projectile Velocity"
LOCATE 15, 55
PRINT "(km/sec)"
BoxRow% = 13
BoxColumn% = 69
BoxLength% = 10
GOSUB BoxDraw
',
'field 12
',
LOCATE 17, 1
PRINT "Bumper Hole Size -"
LOCATE 17, 21
PRINT "Major Axis"
LOCATE 18, 24
PRINT "(in)"
BoxRow% = 16
BoxColumn% = 32
BoxLength% = 10
GOSUB BoxDraw
',
'field 13
',
LOCATE 17, 58
PRINT "Minor Axis"
LOCATE 18, 61
PRINT "(in)"
BoxRow% = 16
BoxColumn% = 69
BoxLength% = 10
GOSUB BoxDraw
',
'field 14
',
LOCATE 20, 1
PRINT "MLI Hole Diameter"
LOCATE 21, 8
PRINT "(in)"
BoxRow% = 19
BoxColumn% = 19
BoxLength% = 10
GOSUB BoxDraw
',
```

DATABASE.BAS

```
'field 15
,
LOCATE 20, 55
PRINT "MLI Mass Loss"
LOCATE 21, 58
PRINT "(grams)"
BoxRow% = 19
BoxColumn% = 69
BoxLength% = 10
GOSUB BoxDraw
,
'field 16
,
LOCATE 23, 1
PRINT "Pressure Wall Hole Size - Major Axis"
LOCATE 24, 30
PRINT "(in)";
BoxRow% = 22
BoxColumn% = 38
BoxLength% = 10
GOSUB BoxDraw
,
'field 17
,
LOCATE 23, 58
PRINT "Minor Axis"
LOCATE 24, 61
PRINT "(in)";
BoxRow% = 22
BoxColumn% = 69
BoxLength% = 10
GOSUB BoxDraw
RETURN
BoxDraw:
COLOR 9, 0
LOCATE BoxRow%, BoxColumn%
BoxLine$ = CHR$(201)
FOR i% = 1 TO BoxLength%
BoxLine$ = BoxLine$ + CHR$(205)
NEXT i%
BoxLine$ = BoxLine$ + CHR$(187)
PRINT BoxLine$;
LOCATE BoxRow% + 1, BoxColumn%
BoxLine$ = CHR$(186)
PRINT BoxLine$;
LOCATE BoxRow% + 1, BoxColumn% + 1 + BoxLength%
BoxLine$ = CHR$(186)
PRINT BoxLine$;
LOCATE BoxRow% + 2, BoxColumn%
BoxLine$ = CHR$(200)
FOR i% = 1 TO BoxLength%
BoxLine$ = BoxLine$ + CHR$(205)
NEXT i%
BoxLine$ = BoxLine$ + CHR$(188)
PRINT BoxLine$;
COLOR 11, 0
RETURN
WriteDataToFile:
PRINT #1, "{"
PRINT #1, TestID$
```


DATABASE.BAS

```
PRINT #1, DataSource$
PRINT #1, TestDate$
PRINT #1, BumperMaterial$
PRINT #1, BumperThickness#
PRINT #1, BumperStandOff#
PRINT #1, PressureWallMaterial$
PRINT #1, PressureWallThickness#
PRINT #1, ProjectileMaterial$
PRINT #1, ProjectileDiameter#
PRINT #1, ImpactAngle#
PRINT #1, ProjectileVelocity#
PRINT #1, BumperMajorAxis#
PRINT #1, BumperMinorAxis#
PRINT #1, MLIHoleDiam#
PRINT #1, MLIMassLoss#
PRINT #1, PressWallMajAxis#
PRINT #1, PressWallMinAxis#
PRINT #1, "}"
RETURN
DataInputProblem:
GOTO AddData1
RETURN
```

INVRMETH.BAS

```
'  
'  
' INVRMETH.BAS  
' Source code for the inverse R method damage prediction program  
'  
' This program predicts hypervelocity impact damage using the 1/R^(N-1)  
' interpolation/extrapolation scheme described in the MLIBlast Manual.  
' Dimensioned for 100 data points and 50 active degrees of freedom.  
'  
'  
DIM ActiveDOF%(1 TO 50)  
DIM TestID$(1 TO 100)  
DIM DataSource$(1 TO 100)  
DIM TestDate$(1 TO 100)  
DIM MaterialDataTypes$(1 TO 11)  
'Dimensioned for 10 material property attributes for each material.  
DIM BumperMaterial#(1 TO 100, 1 TO 10)  
'Store the average of the material property attributes for the bumper.  
DIM BumperMaterialAve#(1 TO 10)  
DIM BumperThickness#(1 TO 100)  
DIM BumperStandOff#(1 TO 100)  
DIM PressureWallMaterial#(1 TO 100, 1 TO 10)  
DIM PressureWallMaterialAve#(1 TO 10)  
DIM PressureWallThickness#(1 TO 100)  
DIM ProjectileMaterial#(1 TO 100, 1 TO 10)  
DIM ProjectileMaterialAve#(1 TO 10)  
DIM ProjectileDiameter#(1 TO 100)  
DIM ImpactAngle#(1 TO 100)  
DIM ProjectileVelocity#(1 TO 100)  
DIM BumperMajorAxis#(1 TO 100)  
DIM BumperMinorAxis#(1 TO 100)  
DIM MLIHoleDiam#(1 TO 100)  
DIM MLIMassLoss#(1 TO 100)  
DIM PressWallMajAxis#(1 TO 100)  
DIM PressWallMinAxis#(1 TO 100)  
DIM Material#(1 TO 10)  
DIM PredictBumpMat#(1 TO 10)  
DIM PredictPressWallMat#(1 TO 10)  
DIM PredictProjMat#(1 TO 10)  
DIM Xcalc#(1 TO 10)  
DIM Fcalc#(1 TO 10)  
DIM Weight#(1 TO 10)  
DIM a#(1 TO 5, 1 TO 5)  
DIM B#(1 TO 5)  
DIM c#(1 TO 5)  
' Set up screen.  
COLOR 9, 0  
CLS  
LOCATE 1, 1  
PRINT CHR$(201);  
FOR i% = 1 TO 78  
IF i% <> 39 THEN PRINT CHR$(205);  
IF i% = 39 THEN PRINT CHR$(203);  
NEXT i%  
PRINT CHR$(187)  
LOCATE 2, 1: PRINT CHR$(186)  
LOCATE 2, 40: PRINT CHR$(186)  
LOCATE 2, 80: PRINT CHR$(186)  
LOCATE 3, 1  
PRINT CHR$(200);
```

INVRMETH.BAS

```

FOR i% = 1 TO 78
IF i% <> 39 THEN PRINT CHR$(205);
IF i% = 39 THEN PRINT CHR$(202);
NEXT i%
PRINT CHR$(188)
ON ERROR GOTO TestDataFileError 'Trap input test data file name errors.
LOCATE 2, 3
row% = 2
col% = 24
COLOR 11, 0
PRINT "Test Data File Name? ";
COLOR 12, 0
INPUT "", MLITestDataFile$
OPEN MLITestDataFile$ FOR INPUT AS #1
ON ERROR GOTO MaterialDataFileError 'Trap input material data file name errors.
LOCATE 2, 42
row% = 2
col% = 67
COLOR 11, 0
PRINT "Material Data File Name? ";
COLOR 12, 0
INPUT "", MaterialDataFile$
OPEN MaterialDataFile$ FOR INPUT AS #2
ON ERROR GOTO 0
' Count how many data values are given for each material type.
' The user can include 10 properties on each material type.
NumMatDat% = 0 'Stores the number of data values for each material type.
NumMatDat1:
INPUT #2, MaterialDataTypes$(NumMatDat% + 1)
IF MaterialDataTypes$(NumMatDat% + 1) = "{" GOTO NumMatDat2:
NumMatDat% = NumMatDat% + 1
GOTO NumMatDat1
NumMatDat2:
CLOSE #2
NumData% = 0 'Number of data records in the database file.
' Intialize all the average values variables to zero.
BumpThkAve# = 0
BumpStandOffAve# = 0
PressWallThkAve# = 0
ProjDiaAve# = 0
ImpAngAve# = 0
ProjVelAve# = 0
BumpMajAxisAve# = 0
BumpMinAxisAve# = 0
MLIHoleDiamAve# = 0
MLIMassLossAve# = 0
PressWallMajAxisAve# = 0
PressWallMinAxisAve# = 0
'Modify top line of boarder.
COLOR 9, 0
LOCATE 3, 1
PRINT CHR$(204)
LOCATE 3, 80
PRINT CHR$(185)
VIEW PRINT 4 TO 12
' Set row% to -1 to trap material name not found type of error in test data fil
row% = -1
DO WHILE NOT EOF(1)
    NumData% = NumData% + 1
    INPUT #1, dummy$

```

INVRMETH.BAS

```

INPUT #1, TestID$(NumData%)
INPUT #1, DataSource$(NumData%)
INPUT #1, TestDate$(NumData%)
COLOR 9, 0
PRINT CHR$(186);
COLOR 11, 0
PRINT " No. : ";
PRINT NumData%;
PRINT "   ID: ";
PRINT TestID$(NumData%);
PRINT "   Source: ";
PRINT DataSource$(NumData%);
PRINT "   Date: ";
PRINT TestDate$(NumData%);
LOCATE CSRLIN, 80
COLOR 9, 0
PRINT CHR$(186)
' Input bumper material properties.
INPUT #1, MaterialID$
GOSUB GetMaterialProp 'Read in material properties associated with material n
FOR i% = 1 TO NumMatDat%
    BumperMaterial#(NumData%, i%) = Material#(i%)
NEXT i%
INPUT #1, BumperThickness$(NumData%)
BumpThkAve# = BumpThkAve# + BumperThickness$(NumData%)
INPUT #1, BumperStandOff$(NumData%)
BumpStandOffAve# = BumpStandOffAve# + BumperStandOff$(NumData%)
' Input pressure wall material properties.
INPUT #1, MaterialID$
GOSUB GetMaterialProp
FOR i% = 1 TO NumMatDat%
    PressureWallMaterial#(NumData%, i%) = Material#(i%)
NEXT i%
INPUT #1, PressureWallThickness$(NumData%)
PressWallThkAve# = PressWallThkAve# + PressureWallThickness$(NumData%)
' Input projectile material properties.
INPUT #1, MaterialID$
GOSUB GetMaterialProp
FOR i% = 1 TO NumMatDat%
    ProjectileMaterial#(NumData%, i%) = Material#(i%)
NEXT i%
INPUT #1, ProjectileDiameter$(NumData%)
ProjDiaAve# = ProjDiaAve# + ProjectileDiameter$(NumData%)
INPUT #1, ImpactAngle$(NumData%)
ImpAngAve# = ImpAngAve# + ImpactAngle$(NumData%)
INPUT #1, ProjectileVelocity$(NumData%)
ProjVelAve# = ProjVelAve# + ProjectileVelocity$(NumData%)
INPUT #1, BumperMajorAxis$(NumData%)
BumpMajAxisAve# = BumpMajAxisAve# + BumperMajorAxis$(NumData%)
INPUT #1, BumperMinorAxis$(NumData%)
BumpMinAxisAve# = BumpMinAxisAve# + BumperMinorAxis$(NumData%)
INPUT #1, MLIHoleDiam$(NumData%)
MLIHoleDiamAve# = MLIHoleDiamAve# + MLIHoleDiam$(NumData%)
INPUT #1, MLIMassLoss$(NumData%)
MLIMassLossAve# = MLIMassLossAve# + MLIMassLoss$(NumData%)
INPUT #1, PressWallMajAxis$(NumData%)
PressWallMajAxisAve# = PressWallMajAxisAve# + PressWallMajAxis$(NumData%)
INPUT #1, PressWallMinAxis$(NumData%)
PressWallMinAxisAve# = PressWallMinAxisAve# + PressWallMinAxis$(NumData%)
INPUT #1, dummy$

```

INVRMETH.BAS

```

LOOP
VIEW PRINT
'build box for averages
LOCATE 12, 2
FOR i% = 1 TO 78
    PRINT CHR$(205);
NEXT i%
LOCATE 12, 1: PRINT CHR$(204)
LOCATE 12, 40: PRINT CHR$(203)
LOCATE 12, 80: PRINT CHR$(185)
FOR i% = 13 TO 23
    LOCATE i%, 1: PRINT CHR$(186)
    LOCATE i%, 40: PRINT CHR$(186)
    LOCATE i%, 80: PRINT CHR$(186)
NEXT i%
LOCATE 24, 1: PRINT CHR$(200);
FOR i% = 1 TO 78
    IF i% <> 39 THEN PRINT CHR$(205);
    IF i% = 39 THEN PRINT CHR$(202);
NEXT i%
PRINT CHR$(188)
' Calculate and print out database average values.
BumpThkAve# = BumpThkAve# / NumData%
LOCATE 13, 3
COLOR 11, 0
PRINT "Ave. Bumper Thk (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; BumpThkAve#;
BumpStandOffAve# = BumpStandOffAve# / NumData%
LOCATE 13, 42
COLOR 11, 0
PRINT "Ave. Bump. Stand Off (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; BumpStandOffAve#;
PressWallThkAve# = PressWallThkAve# / NumData%
LOCATE 15, 3
COLOR 11, 0
PRINT "Ave. Pres Wall Thk (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; PressWallThkAve#;
ProjDiaAve# = ProjDiaAve# / NumData%
LOCATE 15, 42
COLOR 11, 0
PRINT "Ave. Proj. Dia. (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; ProjDiaAve#;
ImpAngAve# = ImpAngAve# / NumData%
LOCATE 17, 3
COLOR 11, 0
PRINT "Ave. Impact Angle (deg):";
COLOR 15, 0
PRINT USING "##.###^^^^"; ImpAngAve#;
ProjVelAve# = ProjVelAve# / NumData%
LOCATE 17, 42
COLOR 11, 0
PRINT "Ave. Proj. Vel. (km/sec):";
COLOR 15, 0
PRINT USING "##.###^^^^"; ProjVelAve#;
BumpMajAxisAve# = BumpMajAxisAve# / NumData%
LOCATE 19, 3

```

INVRMETH.BAS

```

COLOR 11, 0
PRINT "Ave. Maj. Bumper Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; BumpMajAxisAve#;
BumpMinAxisAve# = BumpMinAxisAve# / NumData%
LOCATE 19, 42
COLOR 11, 0
PRINT "Ave. Min. Bumper Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; BumpMinAxisAve#;
MLIHoleDiamAve# = MLIHoleDiamAve# / NumData%
LOCATE 21, 3
COLOR 11, 0
PRINT "Ave. MLI Hole Diam. (in):";
COLOR 15, 0
PRINT USING "##.###^"; MLIHoleDiamAve#;
MLIMassLossAve# = MLIMassLossAve# / NumData%
LOCATE 21, 42
COLOR 11, 0
PRINT "Ave. MLI Mass Loss (grams):";
COLOR 15, 0
PRINT USING "##.###^"; MLIMassLossAve#;
PressWallMajAxisAve# = PressWallMajAxisAve# / NumData%
LOCATE 23, 3
COLOR 11, 0
PRINT "Ave. Maj. P.Wall Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; PressWallMajAxisAve#;
PressWallMinAxisAve# = PressWallMinAxisAve# / NumData%
LOCATE 23, 42
COLOR 11, 0
PRINT "Ave. Min. P.Wall Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; PressWallMinAxisAve#;
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = ""          'Press any key to continue
'
' Normalize data with respect to calculated averages and
' determine which DOF are active. ActiveDOF%(i%) = 0 if i-th
' DOF is not active and = 1 if is active (ie - changes in the database).
'
FOR i% = 1 TO 50
    ActiveDOF%(i%) = 0
NEXT i%
FOR i% = 1 TO NumData%
    BumperThickness#(i%) = BumperThickness#(i%) / BumpThkAve#
    IF BumperThickness#(i%) < .999 OR BumperThickness#(i%) > 1.001 THEN ActiveDOF
    BumperStandOff#(i%) = BumperStandOff#(i%) / BumpStandOffAve#
    IF BumperStandOff#(i%) < .999 OR BumperStandOff#(i%) > 1.001 THEN ActiveDOF%
    PressureWallThickness#(i%) = PressureWallThickness#(i%) / PressWallThkAve#
    IF PressureWallThickness#(i%) < .999 OR PressureWallThickness#(i%) > 1.001 TH
    ProjectileDiameter#(i%) = ProjectileDiameter#(i%) / ProjDiaAve#
    IF ProjectileDiameter#(i%) < .999 OR ProjectileDiameter#(i%) > 1.001 THEN Act
    IF ImpAngAve# <> 0 THEN ImpactAngle#(i%) = ImpactAngle#(i%) / ImpAngAve#
    IF ImpactAngle#(i%) < .999 OR ImpactAngle#(i%) > 1.001 THEN ActiveDOF%(5) = 1
    ProjectileVelocity#(i%) = ProjectileVelocity#(i%) / ProjVelAve#
    IF ProjectileVelocity#(i%) < .999 OR ProjectileVelocity#(i%) > 1.001 THEN Act

```

INVRMETH.BAS

```

NEXT i%
' Normalize the material data
FOR i% = 1 TO NumMatDat%
  FOR j% = 1 TO NumData%
    BumperMaterialAve#(i%) = BumperMaterialAve#(i%) + BumperMaterial#(j%, i%)
    PressureWallMaterialAve#(i%) = PressureWallMaterialAve#(i%) + PressureWall
    ProjectileMaterialAve#(i%) = ProjectileMaterialAve#(i%) + ProjectileMateri
  NEXT j%
NEXT i%
FOR i% = 1 TO NumMatDat%
  BumperMaterialAve#(i%) = BumperMaterialAve#(i%) / NumData%
  PressureWallMaterialAve#(i%) = PressureWallMaterialAve#(i%) / NumData%
  ProjectileMaterialAve#(i%) = ProjectileMaterialAve#(i%) / NumData%
NEXT i%
' Display material property information.
CLS
LOCATE 2, 1
COLOR 11, 0
PRINT "          MATERIAL          BUMPER          PRESSURE          PROJECT
PRINT "          PROPERTY          MATERIAL          WALL          MATERI
PRINT "          AVE.          MATERIAL AVE.          AVE
FOR i% = 1 TO NumMatDat%
  LOCATE 2 * i% + 4, 3
  COLOR 11, 0
  PRINT MaterialDataTypes$(i%)
  LOCATE 2 * i% + 4, 33
  COLOR 15, 0
  PRINT USING "##.##^"; BumperMaterialAve#(i%)
  LOCATE 2 * i% + 4, 50
  PRINT USING "##.##^"; PressureWallMaterialAve#(i%)
  LOCATE 2 * i% + 4, 67
  PRINT USING "##.##^"; ProjectileMaterialAve#(i%);
NEXT i%
BottomRow% = CSRLIN
' Draw Box Around The Data
FOR i% = 2 TO 79
  COLOR 9, 0
  LOCATE 1, i%: PRINT CHR$(205)
  LOCATE 5, i%: PRINT CHR$(205)
  LOCATE BottomRow% + 1, i%: PRINT CHR$(205);
NEXT i%
FOR i% = 2 TO BottomRow%
  LOCATE i%, 1: PRINT CHR$(186)
  LOCATE i%, 29: PRINT CHR$(186)
  LOCATE i%, 46: PRINT CHR$(179)
  LOCATE i%, 63: PRINT CHR$(179)
  LOCATE i%, 80: PRINT CHR$(186)
NEXT i%
LOCATE 1, 1: PRINT CHR$(201)
LOCATE 1, 29: PRINT CHR$(203)
LOCATE 1, 46: PRINT CHR$(209)
LOCATE 1, 63: PRINT CHR$(209)
LOCATE 1, 80: PRINT CHR$(187)
LOCATE 5, 1: PRINT CHR$(204)
LOCATE 5, 29: PRINT CHR$(206)
LOCATE 5, 46: PRINT CHR$(216)
LOCATE 5, 63: PRINT CHR$(216)
LOCATE 5, 80: PRINT CHR$(185)
LOCATE BottomRow% + 1, 1: PRINT CHR$(200);
LOCATE BottomRow% + 1, 29: PRINT CHR$(202);

```

INVRMETH.BAS

```

LOCATE BottomRow% + 1, 46: PRINT CHR$(207);
LOCATE BottomRow% + 1, 63: PRINT CHR$(207);
LOCATE BottomRow% + 1, 80: PRINT CHR$(188);
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = ""          'Press any key to continue
' Check which material properties are active in the database.
FOR i% = 1 TO NumMatDat%
  FOR j% = 1 TO NumData%
    BumperMaterial#(j%, i%) = BumperMaterial#(j%, i%) / BumperMaterialAve#(i%)
    IF BumperMaterial#(j%, i%) < .999 OR BumperMaterial#(j%, i%) > 1.001 THEN
      PressureWallMaterial#(j%, i%) = PressureWallMaterial#(j%, i%) / PressureWa
    IF PressureWallMaterial#(j%, i%) < .999 OR PressureWallMaterial#(j%, i%) >
      ProjectileMaterial#(j%, i%) = ProjectileMaterial#(j%, i%) / ProjectileMate
    IF ProjectileMaterial#(j%, i%) < .999 OR ProjectileMaterial#(j%, i%) > 1.0
  NEXT j%
NEXT i%
NumActiveDOF% = 0
FOR i% = 1 TO 6 + 3 * NumMatDat% 'Count how many degrees of freedom are active.
  NumActiveDOF% = NumActiveDOF% + ActiveDOF%(i%)
NEXT i%
' Set the default values. These can be changed.
BumperMaterialID$ = "6061-T6"
PredictBumpThick# = .04 / BumpThkAve# 'Default values should be divided by the
PredictBumpStandOff# = 4 / BumpStandOffAve#
PressWallMaterialID$ = "2219-T87"
PredictPressWallThick# = .125 / PressWallThkAve#
ProjectileMaterialID$ = "1100"
PredictProjDia# = .313 / ProjDiaAve#
IF ImpAngAve# <> 0 THEN PredictImpAngle# = 45 / ImpAngAve#
PredictProjVel# = 5.3# / ProjVelAve#
'
' Prompt the user for the prediction required.
'
PredictValue:
SCREEN 0
COLOR 15, 1
CLS
LOCATE 1, 3
PRINT "Please Enter The ";
COLOR 12, 1
PRINT "Number ";
COLOR 15, 1
PRINT "Associated With the Desired Action:"
LOCATE 5, 3
COLOR 12, 1
PRINT "1. ";
COLOR 15, 1
PRINT "Predict Bumper Hole Major Axis"
LOCATE 7, 3
COLOR 12, 1
PRINT "2. ";
COLOR 15, 1
PRINT "Predict Bumper Hole Minor Axis"
LOCATE 9, 3
COLOR 12, 1
PRINT "3. ";
COLOR 15, 1

```


INVRMETH.BAS

```

PRINT "Predict MLI Hole Diameter"
LOCATE 11, 3
COLOR 12, 1
PRINT "4. ";
COLOR 15, 1
PRINT "Predict MLI Mass Loss"
LOCATE 13, 3
COLOR 12, 1
PRINT "5. ";
COLOR 15, 1
PRINT "Predict Pressure Wall Hole Major Axis"
LOCATE 15, 3
COLOR 12, 1
PRINT "6. ";
COLOR 15, 1
PRINT "Predict Pressure Wall Hole Minor Axis"
SelectPredictionType:
LOCATE 19, 1
COLOR 12, 1
INPUT PredictionType%
IF PredictionType% < 1 OR PredictionType% > 6 THEN
LOCATE 21, 1
PRINT "Please re-enter choice!"
LOCATE 19, 1
PRINT "
GOTO SelectPredictionType
END IF
'
' Prompt user for data associated with prediction.
'
COLOR 11, 0
CLS
LOCATE 1, 1
PRINT "ENTER DATA FOR DESIRED PREDICTION: "
COLOR 15, 0
LOCATE 3, 1
PRINT "[default values shown in square brackets]"
COLOR 10, 0
LOCATE 5, 1
PRINT "(magnitude relative to database average shown in round brackets)"
LOCATE 8, 1
COLOR 11, 0
PRINT "Bumper Material:";
COLOR 15, 0
PRINT "["; BumperMaterialID$; "]"
LOCATE 8, 40
row% = 8: col% = 40
COLOR 12, 0
INPUT "", dummy$
IF dummy$ = "" THEN
    MaterialID$ = BumperMaterialID$
ELSE
    BumperMaterialID$ = dummy$
    MaterialID$ = dummy$
END IF
MaterialID$ = UCASE$(MaterialID$)
GOSUB GetMaterialProp
BumperMaterialID$ = MaterialID$
MatAve# = 0 'Used to keep track of average material properties.
DOFWarning% = 0 'A parameter used to warn user that a prediction is requested

```

INVRMETH.BAS

```

                'for a DOF that does not vary in the database.
FOR i% = 1 TO NumMatDat%
    PredictBumpMat#(i%) = Material#(i%) / BumperMaterialAve#(i%)
    IF (PredictBumpMat#(i%) < .999 OR PredictBumpMat#(i%) > 1.001) AND ActiveDOF%
        MatAve# = MatAve# + PredictBumpMat#(i%)
NEXT i%
MatAve# = MatAve# / NumMatDat%
LOCATE 8, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; MatAve#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 9, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 10, 1
PredictBumpThick# = PredictBumpThick# * BumpThkAve#
COLOR 11, 0
PRINT "Bumper Thickness (in):";
COLOR 15, 0
PRINT "[";
PRINT USING "##.###^^^^"; PredictBumpThick#;
PRINT "]"
LOCATE 10, 40
COLOR 12, 0
INPUT "", dummy#
IF dummy# <> 0 THEN PredictBumpThick# = dummy#
PredictBumpThick# = PredictBumpThick# / BumpThkAve#
IF (PredictBumpThick# < .999 OR PredictBumpThick# > 1.001) AND ActiveDOF%(1) = 0
LOCATE 10, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictBumpThick#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 11, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
IF PredictionType% >= 5 THEN 'Include the effects of the pressure wall here.
    DOFWarning% = 0
    LOCATE 12, 1
    PredictBumpStandOff# = PredictBumpStandOff# * BumpStandOffAve#
    COLOR 11, 0
    PRINT "Bumper Stand-off (in):";
    COLOR 15, 0
    PRINT "[";
    PRINT USING "##.###^^^^"; PredictBumpStandOff#;
    PRINT "]"
    COLOR 12, 0
    LOCATE 12, 40
    INPUT "", dummy#
    IF dummy# <> 0 THEN PredictBumpStandOff# = dummy#
    PredictBumpStandOff# = PredictBumpStandOff# / BumpStandOffAve#
    IF (PredictBumpStandOff# < .999 OR PredictBumpStandOff# > 1.001) AND ActiveDO
LOCATE 12, 60
COLOR 10, 0

```

```

PRINT "(";
PRINT USING "##.###"; PredictBumpStandOff#;
PRINT ")"
IF DOFWarning% = 1 THEN
  LOCATE 13, 1
  COLOR 12, 0
  PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 14, 1
COLOR 11, 0
PRINT "Pressure Wall Material:";
COLOR 15, 0
PRINT "["; PressWallMaterialID$; "]"
LOCATE 14, 40
row% = 14: col% = 40
COLOR 12, 0
INPUT "", dummy$
IF dummy$ = "" THEN
  MaterialID$ = PressWallMaterialID$
ELSE
  PressWallMaterialID$ = dummy$
  MaterialID$ = dummy$
END IF
MaterialID$ = UCASE$(MaterialID$)
GOSUB GetMaterialProp
PressWallMaterialID$ = MaterialID$
MatAve# = 0
FOR i% = 1 TO NumMatDat%
  PredictPressWallMat#(i%) = Material#(i%) / PressureWallMaterialAve#(i%)
  IF (PredictPressWallMat#(i%) < .999 OR PredictPressWallMat#(i%) > 1.001) A
  MatAve# = MatAve# + PredictPressWallMat#(i%)
NEXT i%
MatAve# = MatAve# / NumMatDat%
LOCATE 14, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; MatAve#;
PRINT ")"
IF DOFWarning% = 1 THEN
  LOCATE 15, 1
  COLOR 12, 0
  PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 16, 1
PredictPressWallThick# = PredictPressWallThick# * PressWallThkAve#
COLOR 11, 0
PRINT "Press. Wall Thick. (in):";
COLOR 15, 0
PRINT "[";
PRINT USING "##.###^^^^"; PredictPressWallThick#;
PRINT "]"
LOCATE 16, 40
COLOR 12, 0
INPUT "", dummy#
IF dummy# <> 0 THEN PredictPressWallThick# = dummy#
PredictPressWallThick# = PredictPressWallThick# / PressWallThkAve#
IF (PredictPressWallThick# < .999 OR PredictPressWallThick# > 1.001) AND Acti
LOCATE 16, 60

```

INVRMETH.BAS

```

COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictPressWallThick#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 17, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
END IF
DOFWarning% = 0
LOCATE 18, 1
COLOR 11, 0
PRINT "Projectile Material:";
COLOR 15, 0
PRINT "["; ProjectileMaterialID$; "]"
LOCATE 18, 40
row% = 18: col% = 40
COLOR 12, 0
INPUT "", dummy$
IF dummy$ = "" THEN
    MaterialID$ = ProjectileMaterialID$
ELSE
    ProjectileMaterialID$ = dummy$
    MaterialID$ = dummy$
END IF
MaterialID$ = UCASE$(MaterialID$)
GOSUB GetMaterialProp
ProjectileMaterialID$ = MaterialID$
MatAve# = 0
FOR i% = 1 TO NumMatDat%
    PredictProjMat#(i%) = Material#(i%) / ProjectileMaterialAve#(i%)
    IF (PredictProjMat#(i%) < .999 OR PredictProjMat#(i%) > 1.001) AND ActiveDOF
        MatAve# = MatAve# + PredictProjMat#(i%)
NEXT i%
MatAve# = MatAve# / NumMatDat%
LOCATE 18, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; MatAve#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 19, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 20, 1
PredictProjDia# = PredictProjDia# * ProjDiaAve#
COLOR 11, 0
PRINT "Projectile Diameter (in):";
COLOR 15, 0
PRINT "[";
PRINT USING "##.###^^^^"; PredictProjDia#;
PRINT "]"
LOCATE 20, 40
COLOR 12, 0
INPUT "", dummy#
IF dummy# <> 0 THEN PredictProjDia# = dummy#
PredictProjDia# = PredictProjDia# / ProjDiaAve#

```

INVRMETH.BAS

```

IF (PredictProjDia# < .999 OR PredictProjDia# > 1.001) AND ActiveDOF%(4) = 0 THE
LOCATE 20, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictProjDia#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 21, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 22, 1
COLOR 11, 0
PRINT "Impact Angle (degrees):";
LOCATE 22, 40
COLOR 12, 0
INPUT "", PredictImpAngle#
IF ImpAngAve# <> 0 THEN PredictImpAngle# = PredictImpAngle# / ImpAngAve#
IF ImpAngAve# = 0 AND PredictImpAngle# > 0 THEN
    LOCATE 23, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
IF ImpAngAve# <> 0 THEN
    IF (PredictImpAngle# < .999 OR PredictImpAngle# > 1.001) AND ActiveDOF%(5) =
    LOCATE 22, 60
    COLOR 10, 0
    PRINT "(";
    PRINT USING "##.###"; PredictImpAngle#;
    PRINT ")"
    IF DOFWarning% = 1 THEN
        LOCATE 23, 1
        COLOR 12, 0
        PRINT "Warning-Your test data does not support the above input data!"
    END IF
END IF
DOFWarning% = 0
LOCATE 24, 1
PredictProjVel# = PredictProjVel# * ProjVelAve#
COLOR 11, 0
PRINT "Proj. Vel. (km/sec):";
COLOR 15, 0
PRINT "[";
PRINT USING "##.###^^^^"; PredictProjVel#;
PRINT "]"
LOCATE 24, 40
COLOR 12, 0
INPUT ; "", dummy#
IF dummy# <> 0 THEN PredictProjVel# = dummy#
PredictProjVel# = PredictProjVel# / ProjVelAve#
IF (PredictProjVel# < .999 OR PredictProjVel# > 1.001) AND ActiveDOF%(6) = 0 THE
LOCATE 24, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictProjVel#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 25, 1
    COLOR 12, 0

```

INVRMETH.BAS

```

PRINT "Warning-Your test data does not support the above input data!";
END IF
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = ""           'Press any key to continue
'
' Find distance of user input point from origin.
'
PredictVectMag# = 0
FOR i% = 1 TO NumMatDat%
  PredictVectMag# = PredictVectMag# + PredictBumpMat#(i%) ^ 2
  IF PredictionType% >= 5 THEN
    PredictVectMag# = PredictVectMag# + PredictPressWallMat#(i%) ^ 2
  END IF
  PredictVectMag# = PredictVectMag# + PredictProjMat#(i%) ^ 2
NEXT i%
PredictVectMag# = PredictVectMag# + PredictBumpThick# ^ 2
IF PredictionType% >= 5 THEN
  PredictVectMag# = PredictVectMag# + PredictBumpStandOff# ^ 2
  PredictVectMag# = PredictVectMag# + PredictPressWallThick# ^ 2
END IF
PredictVectMag# = PredictVectMag# + PredictProjDia# ^ 2
PredictVectMag# = PredictVectMag# + PredictImpAngle# ^ 2
PredictVectMag# = PredictVectMag# + PredictProjVel# ^ 2
PredictVectMag# = PredictVectMag# ^ .5
' Determine near and far points of data points along user input prediction vect
Far# = 0
DistanceAve# = 0
FOR i% = 1 TO NumData%
  Distance# = 0
  FOR j% = 1 TO NumMatDat%
    Distance# = Distance# + PredictBumpMat#(j%) * BumperMaterial#(i%, j%)
    IF PredictionType% >= 5 THEN
      Distance# = Distance# + PredictPressWallMat#(j%) * PressureWallMaterial
    END IF
    Distance# = Distance# + PredictProjMat#(j%) * ProjectileMaterial#(i%, j%)
  NEXT j%
  Distance# = Distance# + PredictBumpThick# * BumperThickness#(i%)
  IF PredictionType% >= 5 THEN
    Distance# = Distance# + PredictBumpStandOff# * BumperStandOff#(i%)
    Distance# = Distance# + PredictPressWallThick# * PressureWallThickness#(i%)
  END IF
  Distance# = Distance# + PredictProjDia# * ProjectileDiameter#(i%)
  Distance# = Distance# + PredictImpAngle# * ImpactAngle#(i%)
  Distance# = Distance# + PredictProjVel# * ProjectileVelocity#(i%)
  Distance# = Distance# / PredictVectMag#
  IF Far# = 0 THEN
    Near# = Distance#
    Far# = Distance#
  ELSE
    IF Distance# < Near# THEN Near# = Distance#
    IF Distance# > Far# THEN Far# = Distance#
  END IF
  DistanceAve# = DistanceAve# + Distance#
NEXT i%
DistanceAve# = DistanceAve# / NumData%
SCREEN 0
COLOR 9, 0

```

NVRMETH.BAS

```

CLS
LOCATE 1, 1: PRINT CHR$(201)
FOR i% = 2 TO 79
LOCATE 1, i%: PRINT CHR$(205)
NEXT i%
LOCATE 1, 80: PRINT CHR$(187)
Relative& = 100 * PredictVectMag# / DistanceAve#
PRINT CHR$(186);
COLOR 11, 0
PRINT " Relative distance of desired prediction from origin: ";
COLOR 15, 0
PRINT Relative&
COLOR 9, 0
LOCATE 2, 80: PRINT CHR$(186)
LOCATE 3, 1: PRINT CHR$(186): LOCATE 3, 80: PRINT CHR$(186)
Relative& = 100
PRINT CHR$(186);
COLOR 11, 0
PRINT " Relative distance of mean data point from the origin: "; Relative&
COLOR 9, 0
LOCATE 4, 80: PRINT CHR$(186)
LOCATE 5, 1: PRINT CHR$(186): LOCATE 5, 80: PRINT CHR$(186)
Relative& = 100 * Far# / DistanceAve#
PRINT CHR$(186);
COLOR 11, 0
PRINT " Relative dist of farthest data point from the origin: "; Relative&
COLOR 9, 0
LOCATE 6, 80: PRINT CHR$(186)
LOCATE 7, 1: PRINT CHR$(186): LOCATE 7, 80: PRINT CHR$(186)
Relative& = 100 * Near# / DistanceAve#
PRINT CHR$(186);
COLOR 11, 0
PRINT " Relative dist of nearest data point from the origin: "; Relative&
COLOR 9, 0
LOCATE 8, 80: PRINT CHR$(186)
LOCATE 9, 1: PRINT CHR$(204)
FOR i% = 2 TO 79
    LOCATE 9, i%: PRINT CHR$(205)
NEXT i%
LOCATE 9, 80: PRINT CHR$(185)
' Calculate ten data points along user input vector
FOR i% = 1 TO 10
    Normalize# = 0
    Delta# = (Far# - Near#) / 20#
    Factor# = (((i% - 1) * 2# + 1#) * Delta# + Near#) / PredictVectMag#
    Xcalc#(i%) = Factor# * PredictVectMag#
    Fcalc#(i%) = 0
    FOR j% = 1 TO NumData%
        r# = 0
        FOR k% = 1 TO NumMatDat%
            r# = r# + (BumperMaterial#(j%, k%) - PredictBumpMat#(k%) * Factor#) ^ 2
            IF PredictionType% >= 5 THEN
                r# = r# + (PressureWallMaterial#(j%, k%) - PredictPressWallMat#(k%)
            END IF
            r# = r# + (ProjectileMaterial#(j%, k%) - PredictProjMat#(k%) * Factor#)
        NEXT k%
        r# = r# + (BumperThickness#(j%) - PredictBumpThick# * Factor#) ^ 2
        IF PredictionType% >= 5 THEN
            r# = r# + (BumperStandOff#(j%) - PredictBumpStandOff# * Factor#) ^ 2
            r# = r# + (PressureWallThickness#(j%) - PredictPressWallThick# * Factor

```

INVRMETH.BAS

```

    END IF
    r# = r# + (ProjectileDiameter#(j%) - PredictProjDia# * Factor#) ^ 2
    r# = r# + (ImpactAngle#(j%) - PredictImpAngle# * Factor#) ^ 2
    r# = r# + (ProjectileVelocity#(j%) - PredictProjVel# * Factor#) ^ 2
    r# = r# ^ .5
    Normalize# = Normalize# + 1# / r# ^ NumActiveDOF%
    IF PredictionType% = 1 THEN Fcalc#(i%) = Fcalc#(i%) + BumperMajorAxis#(j%)
    IF PredictionType% = 2 THEN Fcalc#(i%) = Fcalc#(i%) + BumperMinorAxis#(j%)
    IF PredictionType% = 3 THEN Fcalc#(i%) = Fcalc#(i%) + MLIHoleDiam#(j%) /
    IF PredictionType% = 4 THEN Fcalc#(i%) = Fcalc#(i%) + MLIMassLoss#(j%) /
    IF PredictionType% = 5 THEN Fcalc#(i%) = Fcalc#(i%) + PressWallMajAxis#(j
    IF PredictionType% = 6 THEN Fcalc#(i%) = Fcalc#(i%) + PressWallMinAxis#(j
NEXT j%
IF Normalize# <> 0 THEN Fcalc#(i%) = Fcalc#(i%) / Normalize#
NEXT i%
'
' Here we fit a polynomial of order Order% through interpolated data points.
Order% = 5
NumGoodData% = 0
sumx# = 0
sumx2# = 0
sumx3# = 0
sumx4# = 0
sumx5# = 0
sumx6# = 0
sumx7# = 0
sumx8# = 0
ybar# = 0
sumy# = 0
sumyx# = 0
sumyx2# = 0
sumyx3# = 0
sumyx4# = 0
SumWeight# = 0
FOR i% = 1 TO 10
    Weight#(i%) = 1# 'Here a provision is made for weighting the data.
    SumWeight# = SumWeight# + Weight#(i%)
    sumx# = sumx# + Xcalc#(i%) * Weight#(i%)
    sumx2# = sumx2# + Xcalc#(i%) ^ 2 * Weight#(i%)
    sumx3# = sumx3# + Xcalc#(i%) ^ 3 * Weight#(i%)
    sumx4# = sumx4# + Xcalc#(i%) ^ 4 * Weight#(i%)
    sumx5# = sumx5# + Xcalc#(i%) ^ 5 * Weight#(i%)
    sumx6# = sumx6# + Xcalc#(i%) ^ 6 * Weight#(i%)
    sumx7# = sumx7# + Xcalc#(i%) ^ 7 * Weight#(i%)
    sumx8# = sumx8# + Xcalc#(i%) ^ 8 * Weight#(i%)
    ybar# = ybar# + Fcalc#(i%)
    sumy# = sumy# + Fcalc#(i%) * Weight#(i%)
    sumyx# = sumyx# + Fcalc#(i%) * Xcalc#(i%) * Weight#(i%)
    sumyx2# = sumyx2# + Fcalc#(i%) * Xcalc#(i%) ^ 2 * Weight#(i%)
    sumyx3# = sumyx3# + Fcalc#(i%) * Xcalc#(i%) ^ 3 * Weight#(i%)
    sumyx4# = sumyx4# + Fcalc#(i%) * Xcalc#(i%) ^ 4 * Weight#(i%)
NEXT i%
' Here the least square equation [a]{b} = {c} is set up and solved.
' {b} are the polynomial coefficients.
ybar# = ybar# / 10
a#(1, 1) = SumWeight#
a#(1, 2) = sumx#
a#(1, 3) = sumx2#
a#(1, 4) = sumx3#
a#(1, 5) = sumx4#

```


INVRMETH.BAS

```

c#(1) = sumy#
a#(2, 1) = sumx#
a#(2, 2) = sumx2#
a#(2, 3) = sumx3#
a#(2, 4) = sumx4#
a#(2, 5) = sumx5#
c#(2) = sumyx#
a#(3, 1) = sumx2#
a#(3, 2) = sumx3#
a#(3, 3) = sumx4#
a#(3, 4) = sumx5#
a#(3, 5) = sumx6#
c#(3) = sumyx2#
a#(4, 1) = sumx3#
a#(4, 2) = sumx4#
a#(4, 3) = sumx5#
a#(4, 4) = sumx6#
a#(4, 5) = sumx7#
c#(4) = sumyx3#
a#(5, 1) = sumx4#
a#(5, 2) = sumx5#
a#(5, 3) = sumx6#
a#(5, 4) = sumx7#
a#(5, 5) = sumx8#
c#(5) = sumyx4#
FOR i% = 1 TO Order% - 1
  FOR j% = i% + 1 TO Order%
    IF a#(j%, i%) <= a#(i%, i%) THEN GOTO L1
    FOR k% = i% TO Order%
      dumb# = a#(i%, k%)
      a#(i%, k%) = a#(j%, k%)
      a#(j%, k%) = dumb#
    NEXT k%
    dumb# = c#(i%)
    c#(i%) = c#(j%)
    c#(j%) = dumb#
  L1:
  NEXT j%
  FOR j% = i% + 1 TO Order%
    Factor# = a#(j%, i%) / a#(i%, i%)
    FOR k% = i% TO Order%
      a#(j%, k%) = a#(j%, k%) - a#(i%, k%) * Factor#
    NEXT k%
    c#(j%) = c#(j%) - c#(i%) * Factor#
  NEXT j%
NEXT i%
FOR i% = Order% TO 1 STEP -1
  sum# = 0
  IF i% = Order% THEN
    B#(i%) = (c#(i%) - sum#) / a#(i%, i%)
    GOTO L2
  END IF
  FOR j% = i% + 1 TO Order%
    sum# = sum# + a#(i%, j%) * B#(j%)
  NEXT j%
  B#(i%) = (c#(i%) - sum#) / a#(i%, i%)
L2:
NEXT i%
sumerr# = 0
sumdif# = 0

```

INVRMETH.BAS

```

FOR i% = 1 TO 10
    fest# = B#(1) + B#(2) * Xcalc#(i%) + B#(3) * Xcalc#(i%) ^ 2 + B#(4) * Xcalc#(
    sumerr# = sumerr# + (Fcalc#(i%) - fest#) ^ 2 * Weight#(i%)
    sumdif# = sumdif# + (Fcalc#(i%) - ybar#) ^ 2 * Weight#(i%)
NEXT i%
IF sumdif# <> 0 THEN CoefDet# = 1 - (sumerr# / sumdif#)
IF sumdif# = 0 THEN CoefDet# = 1
LOCATE 10, 1
COLOR 9, 0
PRINT CHR$(186);
COLOR 11, 0
PRINT " LEAST SQUARES POLYNOMIAL FIT THROUGH TEN INTERPOLATED DATA POINTS:"
COLOR 9, 0
LOCATE 10, 80: PRINT CHR$(186)
COLOR 11, 0
LOCATE 11, 3: PRINT "("; NumData%; " Data Records From ";
COLOR 15, 0
PRINT MLITestDataFile$;
COLOR 11, 0
PRINT " Database Were Used For Interpolation)"
COLOR 9, 0
LOCATE 11, 1: PRINT CHR$(186): LOCATE 11, 80: PRINT CHR$(186)
PRINT CHR$(186);
COLOR 11, 0
PRINT " constant term ";
PRINT USING "##.###^"; B#(1)
COLOR 9, 0
LOCATE 12, 80: PRINT CHR$(186)
PRINT CHR$(186);
COLOR 11, 0
PRINT " x coefficient ";
PRINT USING "##.###^"; B#(2)
COLOR 9, 0
LOCATE 13, 80: PRINT CHR$(186)
PRINT CHR$(186);
COLOR 11, 0
PRINT " x^2 coefficient ";
PRINT USING "##.###^"; B#(3)
COLOR 9, 0
LOCATE 14, 80: PRINT CHR$(186)
PRINT CHR$(186);
COLOR 11, 0
PRINT " x^3 coefficient ";
PRINT USING "##.###^"; B#(4)
COLOR 9, 0
LOCATE 15, 80: PRINT CHR$(186)
PRINT CHR$(186);
COLOR 11, 0
PRINT " x^4 coefficient ";
PRINT USING "##.###^"; B#(5)
COLOR 9, 0
LOCATE 16, 80: PRINT CHR$(186)
PRINT CHR$(186);
COLOR 11, 0
PRINT " POLYNOMIAL coefficient of determination (R^2) = ";
COLOR 15, 0
PRINT USING "##.###^"; CoefDet#
COLOR 9, 0
LOCATE 17, 80: PRINT CHR$(186)
LOCATE 18, 1: PRINT CHR$(204)

```

INVRMETH.BAS

```

FOR i% = 2 TO 79
  LOCATE 18, i%: PRINT CHR$(205)
NEXT i%
LOCATE 18, 80: PRINT CHR$(185)
LOCATE 19, 1: PRINT CHR$(186): LOCATE 19, 80: PRINT CHR$(186)
PREDICTION# = B#(1) + B#(2) * PredictVectMag# + B#(3) * PredictVectMag# ^ 2 + B#
IF PREDICTION# < 0 THEN PREDICTION# = 0
PRINT CHR$(186);
COLOR 11, 0
IF PredictionType% = 1 THEN PRINT " Predicted Bumper Hole Major Axis (in) =";
IF PredictionType% = 2 THEN PRINT " Predicted Bumper Hole Minor Axis (in) =";
IF PredictionType% = 3 THEN PRINT " Predicted MLI Hole Diameter (in) =";
IF PredictionType% = 4 THEN PRINT " Predicted MLI Mass Loss (grams) =";
IF PredictionType% = 5 THEN PRINT " Predicted Pressure Wall Hole Major Axis (in)
IF PredictionType% = 6 THEN PRINT " Predicted Pressure Wall Hole Minor Axis (in)
COLOR 15, 0
PRINT USING "##.###^ ^ ^"; PREDICTION#
COLOR 9, 0
LOCATE 20, 80: PRINT CHR$(186)
LOCATE 21, 1: PRINT CHR$(186): LOCATE 21, 80: PRINT CHR$(186)
LOCATE 22, 1: PRINT CHR$(200)
FOR i% = 2 TO 79
  LOCATE 22, i%: PRINT CHR$(205)
NEXT i%
LOCATE 22, 80: PRINT CHR$(188)
LOCATE 24, 15
COLOR 9, 0
PRINT "Do you wish to see a plot of the results (y/n)? ";
COLOR 12, 0
INPUT "", Answer$
COLOR 15, 0
Answer$ = LCASE$(Answer$)
IF Answer$ = "y" THEN GOSUB Graphics
SCREEN 0
CLS
LOCATE 1, 1
COLOR 9, 0
PRINT "Do You Wish To Enter Data For Another Prediction (y/n)? ";
COLOR 12, 0
INPUT "", Answer$
COLOR 15, 0
Answer$ = LCASE$(Answer$)
IF Answer$ = "y" THEN GOTO PredictValue
END
'
' End of main program.
'
GetMaterialProp: 'This subroutine searches for material property data records
                  'in the material data file.
GetMaterialProp1:
OPEN MaterialDataFile$ FOR INPUT AS #2
FOR mat% = 1 TO NumMatDat%
  INPUT #2, dummy$
NEXT mat%
DO WHILE NOT EOF(2)
  INPUT #2, dummy$
  INPUT #2, TestMaterialID$
  IF TestMaterialID$ = MaterialID$ THEN
    FOR mat% = 1 TO NumMatDat%
      INPUT #2, Material#(mat%)
    
```

INVRMETH.BAS

```

    NEXT mat%
    CLOSE #2
    RETURN
ELSE
    FOR mat% = 1 TO NumMatDat% + 1
        INPUT #2, dummy$
    NEXT mat%
END IF
LOOP
IF row% = -1 THEN
    SCREEN 0
    VIEW PRINT
    COLOR 15, 0
    CLS
    LOCATE 3, 1
    PRINT "ERROR - File ", MLITestDataFile$
    PRINT "References Material Name ", MaterialID$
    PRINT "That Is Not Contained In File ", MaterialDataFile$
    PRINT
    PRINT "press any key to stop";
    DO
    LOOP WHILE INKEY$ = ""           'Press any key to continue
    END
END IF
CLOSE #2
COLOR 12, 0
LOCATE row% + 1, 1
PRINT "Material Name Not Found Please Re-enter (or enter QUIT to stop)"
LOCATE row%, col%
PRINT " "
LOCATE row%, col%
INPUT "", MaterialID$
MaterialID$ = UCASE$(MaterialID$)
IF MaterialID$ = "QUIT" THEN END
LOCATE row% + 1, 1
PRINT " "
GOTO GetMaterialProp1
RETURN
'
' This subroutine graphically displays the results.
'
Graphics:
SCREEN 9
COLOR 15, 1
CLS
' Find largest function value and smallest and largest x values
Fmax! = PREDICTION#
Xmin! = PredictVectMag#
Xmax! = PredictVectMag#
FOR i% = 1 TO 10
    IF Fcalc#(i%) > Fmax! THEN Fmax! = Fcalc#(i%)
    IF Xcalc#(i%) < Xmin! THEN Xmin! = Xcalc#(i%)
    IF Xcalc#(i%) > Xmax! THEN Xmax! = Xcalc#(i%)
NEXT i%
Dx! = Xmax! - Xmin!
w! = Dx! * .01
WINDOW (-Dx! * .7, -Fmax! * .17)-(Dx! * 1.2, Fmax! * 1.2)
LINE (-w! * 5, 0)-(Dx! * 1.1, 0), 7
LINE (-w! * 5, 0)-(-w! * 5, Fmax! * 1.1), 7
LINE (-Dx! * .1, -Fmax! * .07)-(Dx! * 1.16, Fmax! * 1.16), 7, B

```

INVRMETH.BAS

```

FOR i% = 1 TO 10
  LINE (Xcalc#(i%) - Xmin! - w!, 0)-(Xcalc#(i%) - Xmin! + w!, Fcalc#(i%)), 11,
NEXT i%
,
'Display Interpolated Function
COLOR 14, 1
FOR i% = 1 TO 1001
  x# = Xmin! + (i% - 1) * Dx! / 1000#
  f# = B#(1) + B#(2) * x# + B#(3) * x# ^ 2 + B#(4) * x# ^ 3 + B#(5) * x# ^ 4
  IF f# < 0 THEN f# = 0
  PSET (x# - Xmin!, f#)
NEXT i%
,
LINE (PredictVectMag# - Xmin! - w!, -Fmax! * .005)-(PredictVectMag# - Xmin! + w!
LOCATE 2, 30
COLOR 7, 1
PRINT "FUNCTION"
LOCATE 3, 30
PRINT "MAGNITUDE"
LOCATE 23, 56
PRINT " DISTANCE FROM ORIGIN ";
COLOR 11, 1
LOCATE 3, 2
PRINT "      BARS INDICATE"
LOCATE 4, 2
PRINT "DATABASE INTERPOLATIONS"
LINE (-Dx! * .68, Fmax! * 1.05)-(-Dx! * .58, Fmax! * 1.08), 11, BF
LOCATE 7, 2
COLOR 12, 1
PRINT "      BAR INDICATES"
LOCATE 8, 2
PRINT "FUNCTION PREDICTION"
LINE (-Dx! * .68, Fmax! * .83)-(-Dx! * .58, Fmax! * .86), 12, BF
LOCATE 11, 2
COLOR 14, 1
PRINT "      LINE INDICATES"
LOCATE 12, 2
PRINT "LEAST SQUARES FIT"
LOCATE 13, 2
PRINT "THROUGH INTERPOLATIONS"
LINE (-Dx! * .68, Fmax! * .625)-(-Dx! * .58, Fmax! * .63), 14, BF
LOCATE 20, 2
COLOR 7, 1
PRINT "PREDICTED VALUE:"
LOCATE 21, 1
IF PredictionType% = 1 THEN
  PRINT " Bumper Hole"
  PRINT " Major Axis (in) = "
END IF
IF PredictionType% = 2 THEN
  PRINT " Bumper Hole"
  PRINT " Minor Axis (in) = "
END IF
IF PredictionType% = 3 THEN
  PRINT " MLI Hole"
  PRINT " Diameter (in) = "
END IF
IF PredictionType% = 4 THEN
  PRINT " MLI Mass"

```

INVRMETH.BAS

```
PRINT " Loss (grams) ="
END IF
IF PredictionType% = 5 THEN
PRINT " Pressure Wall Hole"
PRINT " Major Axis (in) ="
END IF
IF PredictionType% = 6 THEN
PRINT " Pressure Wall Hole"
PRINT " Minor Axis (in) = "
END IF
LOCATE 23, 4
COLOR 15, 1
PRINT USING "##.##^"; PREDICTION#
LOCATE 25, 28
COLOR 12, 1
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = "" 'Press any key to continue
RETURN
```

TestDataFileError:

```
COLOR 12, 0
LOCATE 4, 1
PRINT "Please Re-enter File Name (or enter QUIT to stop)"
LOCATE row%, col%
PRINT " "
LOCATE row%, col%
INPUT "", MLITestDataFile$
MLITestDataFile$ = UCASE$(MLITestDataFile$)
IF MLITestDataFile$ = "QUIT" THEN END
LOCATE 4, 1
PRINT " "
RESUME
```

MaterialDataFileError:

```
COLOR 12, 0
LOCATE 4, 1
PRINT "Please Re-enter File Name (or enter QUIT) to stop"
LOCATE row%, col%
PRINT " "
LOCATE row%, col%
INPUT "", MaterialDataFile$
MaterialDataFile$ = UCASE$(MaterialDataFile$)
IF MaterialDataFile$ = "QUIT" THEN END
LOCATE 4, 1
PRINT " "
RESUME
```

POLYMETH.BAS

POLYMETH.BAS

Source code for the polynomial functions damage prediction program

```
ECLARE SUB CalcFullAmatrix (i%, a#(), dx#())
ECLARE SUB CalcSmallAmatrix (i%, a#(), dx#())
ECLARE SUB CalcSmallestAmatrix (i%, a#(), dx#())
```

This program fits a linear polynomial through sets of 16 data records.
The polynomial is of the form:

$$\begin{aligned} \text{DAMAGE} = & d1 + d2*x1 + d3*x2 + d4*x3 + d5*x4 \\ & + d6*x1*x2 + d7*x1*x3 + d8*x1*x4 + d9*x2*x3 + d10*x2*x4 + d11*x3*x4 \\ & + d12*x1*x2*x3 + d13*x1*x2*x4 + d14*x1*x3*x4 + d15*x2*x3*x4 \\ & + d16*x1*x2*x3*x4 \end{aligned}$$

where: di = i-th coefficient
x1 = bumper thickness
x2 = projectile diameter
x3 = impact angle
x4 = projectile velocity

If not enough data records are available to fit the full function then the program (either automatically or as directed by the user by pressing function keys) will attempt to fit two lower order (also incomplete) polynomials of the following form:

"simple": Damage = $d1 + d2*x1 + d3*x2 + d4*x3 + d5*x4 + d6*x1*x2 + d7*x1*x3 + d8*x1*x4 + d9*x2*x3 + d10*x2*x4$

"simplest": Daamage = $d1 + d2*x1 + d3*x2 + d4*x3 + d5*x4$

This program assumes that x1 to x4 are the only parameters that vary signific

Here the DAMAGE consists of bumper hole major and minor axis, MLI hole diameter and mass loss, and the pressure wall hole major and minor axis.

```
DECLARE SUB weightedprediction ()
DECLARE SUB ParamWarning ()
Dimensioned for 100 data points.
COMMON SHARED prediction#(), meandistance#(), numsolutions%, finalprediction#, n
DIM activedof%(1 TO 4) 'This variable keeps track of which of x1 to x4 vary in t
DIM TestID$(1 TO 100) 'Data record ID.
DIM DataSource$(1 TO 100) 'Data record source.
DIM TestDate$(1 TO 100) 'Date of data record test.
DIM BumperThickness$(1 TO 100)
DIM ProjectileDiameter$(1 TO 100)
DIM ImpactAngle$(1 TO 100)
DIM ProjectileVelocity$(1 TO 100)
DIM BumperMajoraxis$(1 TO 100)
DIM BumperMinorAxis$(1 TO 100)
DIM MLIHoleDiam$(1 TO 100)
DIM MLIMassLoss$(1 TO 100)
DIM PressWallMajAxis$(1 TO 100)
DIM PressWallMinAxis$(1 TO 100)
DIM meandistance$(1 TO 10) 'Mean distance from prediction point to data points
DIM datasort$(1 TO 100) 'Vector to use for data sorting.
DIM dx$(1 TO 4) 'Distance from xi of data record to xi of prediction point, wh
DIM a$(1 TO 16, 1 TO 16) 'Matrix for solving for function coefficients: [a]{b}
DIM b$(1 TO 16) 'Function coefficients.
```

POLYMETH.BAS

```

DIM c#(1 TO 16) 'Measured damage at a data point.
DIM prediction#(1 TO 5) 'The predictions from 5 sets of coefficients are stored
seed% = ((TIMER * 65536) / 86400) - 32768 'Used for generating random numbers.
RANDOMIZE seed%
numsolutions% = 5 'Number of function fits to be calculated.
CALL ParamWarning 'This subroutine warns the user that only four variables are used
CLS
COLOR 9, 0
LOCATE 1, 1
PRINT CHR$(201);
FOR i% = 1 TO 78
    PRINT CHR$(205);
NEXT i%
PRINT CHR$(187)
LOCATE 2, 1: PRINT CHR$(186)
LOCATE 2, 80: PRINT CHR$(186)
LOCATE 3, 1
PRINT CHR$(200);
FOR i% = 1 TO 78
    PRINT CHR$(205);
NEXT i%
PRINT CHR$(188)
ON ERROR GOTO TestDataFileError 'This traps database file name problems.
LOCATE 2, 3
row% = 2
col% = 24
COLOR 11, 0
PRINT "Test Data File Name? ";
COLOR 12, 0
INPUT "", MLITestDataFile$
OPEN MLITestDataFile$ FOR INPUT AS #1
ON ERROR GOTO 0
numdata% = 0 'Number of data records in the database.
' The following variables store the average values of database data.
BumpThkAve# = 0
ProjDiaAve# = 0
ProjVelAve# = 0
ImpAngAve# = 0
BumpMajAxisAve# = 0
BumpMinAxisAve# = 0
MLIHoleDiamAve# = 0
MLIMassLossAve# = 0
PressWallMajAxisAve# = 0
PressWallMinAxisAve# = 0
'Modify top line of boarder.
COLOR 9, 0
LOCATE 3, 1
PRINT CHR$(204)
LOCATE 3, 80
PRINT CHR$(185)
VIEW PRINT 4 TO 12
' Read in data from database file.
DO WHILE NOT EOF(1)
    numdata% = numdata% + 1
    INPUT #1, dummy$
    INPUT #1, TestID$(numdata%)
    INPUT #1, DataSource$(numdata%)
    INPUT #1, TestDate$(numdata%)
    COLOR 9, 0
    PRINT CHR$(186);

```


POLY METH.BAS

```

COLOR 11, 0
' Scroll data to screen as it is read.
PRINT " No.: ";
PRINT numdata%;
PRINT " ID: ";
PRINT TestID$(numdata%);
PRINT " Source: ";
PRINT DataSource$(numdata%);
PRINT " Date: ";
PRINT TestDate$(numdata%);
LOCATE CSRLIN, 80
COLOR 9, 0
PRINT CHR$(186)
' Input bumper material properties.
INPUT #1, dummy$
INPUT #1, BumperThickness$(numdata%)
BumpThkAve# = BumpThkAve# + BumperThickness$(numdata%)
INPUT #1, dummy#
INPUT #1, dummy$
INPUT #1, dummy#
INPUT #1, dummy$
INPUT #1, ProjectileDiameter$(numdata%)
ProjDiaAve# = ProjDiaAve# + ProjectileDiameter$(numdata%)
INPUT #1, ImpactAngle$(numdata%)
ImpAngAve# = ImpAngAve# + ImpactAngle$(numdata%)
INPUT #1, ProjectileVelocity$(numdata%)
ProjVelAve# = ProjVelAve# + ProjectileVelocity$(numdata%)
INPUT #1, BumperMajoraxis$(numdata%)
BumpMajAxisAve# = BumpMajAxisAve# + BumperMajoraxis$(numdata%)
INPUT #1, BumperMinorAxis$(numdata%)
BumpMinAxisAve# = BumpMinAxisAve# + BumperMinorAxis$(numdata%)
INPUT #1, MLIHoleDiam$(numdata%)
MLIHoleDiamAve# = MLIHoleDiamAve# + MLIHoleDiam$(numdata%)
INPUT #1, MLIMassLoss$(numdata%)
MLIMassLossAve# = MLIMassLossAve# + MLIMassLoss$(numdata%)
INPUT #1, PressWallMajAxis$(numdata%)
PressWallMajAxisAve# = PressWallMajAxisAve# + PressWallMajAxis$(numdata%)
INPUT #1, PressWallMinAxis$(numdata%)
PressWallMinAxisAve# = PressWallMinAxisAve# + PressWallMinAxis$(numdata%)
INPUT #1, dummy$
LOOP
VIEW PRINT
'
'build box for averages
LOCATE 12, 2
FOR i% = 1 TO 78
PRINT CHR$(205);
NEXT i%
LOCATE 12, 1: PRINT CHR$(204)
LOCATE 12, 40: PRINT CHR$(203)
LOCATE 12, 80: PRINT CHR$(185)
FOR i% = 13 TO 23
LOCATE i%, 1: PRINT CHR$(186)
LOCATE i%, 40: PRINT CHR$(186)
LOCATE i%, 80: PRINT CHR$(186)
NEXT i%
LOCATE 17, 1: PRINT CHR$(204);
FOR i% = 1 TO 78
IF i% <> 39 THEN PRINT CHR$(205);
IF i% = 39 THEN PRINT CHR$(206);

```

POLYMETH.BAS

```

NEXT i%
PRINT CHR$(185)
LOCATE 24, 1: PRINT CHR$(200);
FOR i% = 1 TO 78
  IF i% <> 39 THEN PRINT CHR$(205);
  IF i% = 39 THEN PRINT CHR$(202);
NEXT i%
' Print data averages to the screen.
PRINT CHR$(188)
BumpThkAve# = BumpThkAve# / numdata%
LOCATE 13, 3
COLOR 11, 0
PRINT "Ave. Bumper Thk (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; BumpThkAve#;
ProjDiaAve# = ProjDiaAve# / numdata%
LOCATE 13, 42
COLOR 11, 0
PRINT "Ave. Proj. Dia. (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; ProjDiaAve#;
ImpAngAve# = ImpAngAve# / numdata%
LOCATE 15, 3
COLOR 11, 0
PRINT "Ave. Impact Angle (deg):";
COLOR 15, 0
PRINT USING "##.###^^^^"; ImpAngAve#;
ProjVelAve# = ProjVelAve# / numdata%
LOCATE 15, 42
COLOR 11, 0
PRINT "Ave. Proj. Vel. (km/sec):";
COLOR 15, 0
PRINT USING "##.###^^^^"; ProjVelAve#;
BumpMajAxisAve# = BumpMajAxisAve# / numdata%
LOCATE 19, 3
COLOR 11, 0
PRINT "Ave. Maj. Bumper Hole (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; BumpMajAxisAve#;
BumpMinAxisAve# = BumpMinAxisAve# / numdata%
LOCATE 19, 42
COLOR 11, 0
PRINT "Ave. Min. Bumper Hole (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; BumpMinAxisAve#;
MLIHoleDiamAve# = MLIHoleDiamAve# / numdata%
LOCATE 21, 3
COLOR 11, 0
PRINT "Ave. MLI Hole Diam. (in):";
COLOR 15, 0
PRINT USING "##.###^^^^"; MLIHoleDiamAve#;
MLIMassLossAve# = MLIMassLossAve# / numdata%
LOCATE 21, 42
COLOR 11, 0
PRINT "Ave. MLI Mass Loss (grams):";
COLOR 15, 0
PRINT USING "##.###^^^^"; MLIMassLossAve#;
PressWallMajAxisAve# = PressWallMajAxisAve# / numdata%
LOCATE 23, 3
COLOR 11, 0

```

OLYMETH.BAS

```

PRINT "Ave. Maj. P.Wall Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; PressWallMajAxisAve#;
PressWallMinAxisAve# = PressWallMinAxisAve# / numdata%
COLOR 23, 42
COLOR 11, 0
PRINT "Ave. Min. P.Wall Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; PressWallMinAxisAve#;
COLOR 25, 28
COLOR 12, 0
PRINT "press any key to continue";
LEAVE

```

Check which DOF are "active" - ie. vary in the database.

If DOF xi is active then activedof%(i) is set equal to 1 (equal to 0 if not a Also, normalize the data wrt the average.

```

FOR i% = 1 TO numdata%
    BumperThickness#(i%) = BumperThickness#(i%) / BumpThkAve#
    IF BumperThickness#(i%) < .999 OR BumperThickness#(i%) > 1.001 THEN activedof
ProjectileDiameter#(i%) = ProjectileDiameter#(i%) / ProjDiaAve#
    IF ProjectileDiameter#(i%) < .999 OR ProjectileDiameter#(i%) > 1.001 THEN act
    IF ImpAngAve# <> 0 THEN
        ImpactAngle#(i%) = ImpactAngle#(i%) / ImpAngAve#
        IF ImpactAngle#(i%) < .999 OR ImpactAngle#(i%) > 1.001 THEN activedof%(3)
    END IF
    ProjectileVelocity#(i%) = ProjectileVelocity#(i%) / ProjVelAve#
    IF ProjectileVelocity#(i%) < .999 OR ProjectileVelocity#(i%) > 1.001 THEN act
NEXT i%
numactivedof% = 0 'This is the total number of active DOF (can range from 1 to 4
FOR i% = 1 TO 4
    numactivedof% = numactivedof% + activedof%(i%)
NEXT i%
numReqRecords% = 2 ^ numactivedof% 'Calculate number of records required
IF numactivedof% = 1 THEN ReqRecords% = 2 'Calculate number of records requir
IF numactivedof% = 2 THEN ReqRecords% = 4
IF numactivedof% = 3 THEN ReqRecords% = 7
IF numactivedof% = 4 THEN ReqRecords% = 11
numReqRecords% = 1 + numactivedof% 'Find smallest allowable number of records r
IF FullFuncReqRecords% > numdata% THEN 'Issue warning and stop if not enough dat
CLS
COLOR 12, 0
PRINT "Warning - at least ";
COLOR 11, 0
PRINT FullFuncReqRecords%;
COLOR 12, 0
PRINT " data records are required to fit the full function!"
PRINT
PRINT "Data file ";
COLOR 11, 0
PRINT MLITestDataFile$;
COLOR 12, 0
PRINT " only has ";
COLOR 11, 0
PRINT numdata%;
COLOR 12, 0
PRINT " records."
IF numdata% >= MinReqRecords% THEN
    COLOR 13, 0
    PRINT

```

ORIGINAL PAGE IS
OF POOR QUALITY

POLYMETH.BAS

```

PRINT "Fitting incomplete polynomial will now be attempted!"
PRINT
PRINT
COLOR 14, 0
PRINT "Press any key to continue."
SLEEP
GOTO SetDefaultValues
END IF
PRINT
PRINT
COLOR 14, 0
PRINT "Press any key to stop."
SLEEP
STOP
END IF
SetDefaultValues: 'Set the default values - these can be changed.
PredictBumpThick# = .04 / BumpThkAve#
PredictProjDia# = .313 / ProjDiaAve#
IF ImpAngAve# <> 0 THEN PredictImpAngle# = 45 / ImpAngAve#
PredictProjVel# = 5.3# / ProjVelAve#
' Top of prediction loop.
predictvalue:
'
' Here the default order of the function is determined. Variable amatrixcode% k.
IF numdata% >= NumRequiredDataRecord% THEN amatrixcode% = 0 'Attempt to fit full
IF numdata% >= ReqRecords% AND numdata% < NumRequiredDataRecord% THEN amatrixcod
IF numdata% >= MinReqRecords% AND numdata% < ReqRecords% THEN amatrixcode% = 2 '
'
COLOR 15, 1
CLS
' Prompt user for the required prediction.
LOCATE 1, 3
PRINT "Please Enter The ";
COLOR 12, 1
PRINT "Number ";
COLOR 15, 1
PRINT "Associated With the Desired Action:"
LOCATE 5, 3
COLOR 12, 1
PRINT "1. ";
COLOR 15, 1
PRINT "Predict Bumper Hole Major Axis"
LOCATE 7, 3
COLOR 12, 1
PRINT "2. ";
COLOR 15, 1
PRINT "Predict Bumper Hole Minor Axis"
LOCATE 9, 3
COLOR 12, 1
PRINT "3. ";
COLOR 15, 1
PRINT "Predict MLI Hole Diameter"
LOCATE 11, 3
COLOR 12, 1
PRINT "4. ";
COLOR 15, 1
PRINT "Predict MLI Mass Loss"
LOCATE 13, 3
COLOR 12, 1
PRINT "5. ";

```

OLYMETH.BAS

```

OLOR 15, 1
RINT "Predict Pressure Wall Hole Major Axis"
LOCATE 15, 3
OLOR 12, 1
RINT "6. ";
OLOR 15, 1
RINT "Predict Pressure Wall Hole Minor Axis"
LOCATE 17, 3
OLOR 12, 1
RINT "7. ";
OLOR 15, 1
RINT "Quit"
SelectPredictionType:
LOCATE 19, 1
OLOR 12, 1
INPUT "Enter Number"; predictiontype%
IF predictiontype% < 1 OR predictiontype% > 7 THEN
    LOCATE 19, 1
    PRINT "
    GOTO SelectPredictionType
END IF
IF predictiontype% = 7 THEN END
OLOR 11, 0
CLS
LOCATE 1, 1
PRINT "ENTER DATA FOR DESIRED PREDICTION: "
OLOR 15, 0
LOCATE 3, 1
PRINT "[default values shown in square brackets]"
OLOR 10, 0
LOCATE 5, 1
PRINT "(magnitude relative to database average shown in round brackets)"
DOFWarning% = 0
LOCATE 8, 1
PredictBumpThick# = PredictBumpThick# * BumpThkAve#
OLOR 11, 0
PRINT "Bumper Thickness (in):";
OLOR 15, 0
PRINT "[";
PRINT USING "##.###^^^"; PredictBumpThick#;
PRINT "]"
LOCATE 8, 40
OLOR 12, 0
INPUT "", dummy#
IF dummy# <> 0 THEN PredictBumpThick# = dummy#
PredictBumpThick# = PredictBumpThick# / BumpThkAve#
IF (PredictBumpThick# < .999 OR PredictBumpThick# > 1.001) AND activedof%(1) = 0
LOCATE 8, 60
OLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictBumpThick#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 10, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 12, 1
PredictProjDia# = PredictProjDia# * ProjDiaAve#

```

ORIGINAL PAGE IS
OF POOR QUALITY

POLYMETH.BAS

```

COLOR 11, 0
PRINT "Projectile Diameter (in):";
COLOR 15, 0
PRINT "[";
PRINT USING "##.###^"; PredictProjDia#;
PRINT "]"
LOCATE 12, 40
COLOR 12, 0
INPUT "", dummy#
IF dummy# <> 0 THEN PredictProjDia# = dummy#
PredictProjDia# = PredictProjDia# / ProjDiaAve#
IF (PredictProjDia# < .999 OR PredictProjDia# > 1.001) AND activedof%(2) = 0 THEN
LOCATE 12, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictProjDia#;
PRINT ")"
IF DOFWarning% = 1 THEN
    LOCATE 14, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 16, 1
COLOR 11, 0
PRINT "Impact Angle (degrees):";
LOCATE 16, 40
COLOR 12, 0
INPUT "", PredictImpAngle#
IF ImpAngAve# <> 0 THEN
    PredictImpAngle# = PredictImpAngle# / ImpAngAve#
    IF (PredictImpAngle# < .999 OR PredictImpAngle# > 1.001) AND activedof%(3) =
LOCATE 16, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictImpAngle#;
PRINT ")"
    IF DOFWarning% = 1 THEN
        LOCATE 18, 1
        COLOR 12, 0
        PRINT "Warning-Your test data does not support the above input data!"
    END IF
END IF
IF ImpAngAve# = 0 AND PredictImpAngle# > 0 THEN
    LOCATE 18, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!"
END IF
DOFWarning% = 0
LOCATE 20, 1
PredictProjVel# = PredictProjVel# * ProjVelAve#
COLOR 11, 0
PRINT "Proj. Vel. (km/sec):";
COLOR 15, 0
PRINT "[";
PRINT USING "##.###^"; PredictProjVel#;
PRINT "]"
LOCATE 20, 40
COLOR 12, 0
INPUT ; "", dummy#

```

POLYMETH.BAS

```

IF dummy# <> 0 THEN PredictProjVel# = dummy#
PredictProjVel# = PredictProjVel# / ProjVelAve#
IF (PredictProjVel# < .999 OR PredictProjVel# > 1.001) AND activedof%(4) = 0 THE
LOCATE 20, 60
COLOR 10, 0
PRINT "(";
PRINT USING "##.###"; PredictProjVel#;
PRINT ")";
IF DOFWarning% = 1 THEN
    LOCATE 22, 1
    COLOR 12, 0
    PRINT "Warning-Your test data does not support the above input data!";
END IF
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
SLEEP
CLS
calculationattempts% = 0 'Stores the number of attempts made to seek function
IF amatrixcode% = 0 THEN
    LOCATE 21, 20: COLOR 10, 0
    PRINT "Currently Seeking Full Function"
END IF
IF amatrixcode% = 1 THEN
    LOCATE 21, 20: COLOR 10, 0
    PRINT "Currently Seeking Simpler Function"
END IF
IF amatrixcode% = 2 THEN
    LOCATE 21, 20: COLOR 10, 0
    PRINT "Currently Seeking Simplest Function"
END IF
LOCATE 22, 22: COLOR 12, 1
PRINT " PRESS F1 TO STOP CALCULATIONS      "
LOCATE 23, 22: COLOR 15, 1
PRINT " PRESS F2 TO TRY SIMPLER FUNCTION  ";
LOCATE 24, 22: COLOR 11, 1
PRINT " PRESS F3 TO TRY SIMPLEST FUNCTION ";
solution% = 1 'Stores the number of solutions obtained.
FindSolution:
' Use function keys to allow user to select the function to be fit.
ON KEY(1) GOSUB stopcalculations 'User can stop calculations if no solutions
KEY(1) ON
ON KEY(2) GOSUB simplifycalculations 'User can request simpler function if no
KEY(2) ON
ON KEY(3) GOSUB simplestcalculations 'User can request simpler function if no
KEY(3) ON
InitialDataSort: 'Here we randomly select data points for the function fit.
FOR i% = 1 TO 2 ^ numactivedof%
tryagain1:
    try% = numdata% * RND + 1
    IF try% > numdata% THEN try% = numdata%
    IF i% > 1 THEN
        FOR j% = 1 TO i% - 1
            IF try% = datasort%(j%) THEN GOTO tryagain1 'Make sure the same data
            NEXT j%
        datasort%(i%) = try%
    ELSE
        datasort%(1) = try%
    END IF
NEXT i%

```

POLYMETH.BAS

```

ON ERROR GOTO RankError 'If selected data points are not linearly independent
LOCATE 20, 20: COLOR 14, 0
PRINT "Working on calculation attempt: ";
LOCATE 20, 55: COLOR 15, 0
PRINT calculationattempts%
CalculateCoefficients:
calculationattempts% = calculationattempts% + 1
LOCATE 20, 55: COLOR 15, 0
PRINT calculationattempts%
COLOR 15, 0
meandistance#(solution%) = 0#
IF amatrixcode% = 0 THEN order% = FullFuncReqRecords% 'Full function.
IF amatrixcode% = 1 THEN order% = ReqRecords% 'Simpler function.
IF amatrixcode% = 2 THEN order% = MinReqRecords% 'Simplest function.
FOR i% = 1 TO order%
  FOR j% = 1 TO 4
    dx#(j%) = 0#
  NEXT j%
  j% = 1 'This counter is used to ensure that only active DOF are placed i
  IF activedof%(1) = 1 THEN
    dx#(j%) = BumperThickness#(datasort%(i%)) - PredictBumpThick#
    j% = j% + 1
  END IF
  IF activedof%(2) = 1 THEN
    dx#(j%) = ProjectileDiameter#(datasort%(i%)) - PredictProjDia#
    j% = j% + 1
  END IF
  IF activedof%(3) = 1 THEN
    dx#(j%) = ImpactAngle#(datasort%(i%)) - PredictImpAngle#
    j% = j% + 1
  END IF
  IF activedof%(4) = 1 THEN
    dx#(j%) = ProjectileVelocity#(datasort%(i%)) - PredictProjVel#
  END IF
  distance# = 0#
  FOR j% = 1 TO 4
    distance# = distance# + dx#(j%) ^ 2
  NEXT j%
  distance# = SQR(distance#) 'Calculate the "distance" in the design space :
  meandistance#(solution%) = meandistance#(solution%) + distance#
  IF amatrixcode% = 0 THEN CALL CalcFullAmatrix(i%, a#(), dx#())
  IF amatrixcode% = 1 THEN CALL CalcSmallAmatrix(i%, a#(), dx#())
  IF amatrixcode% = 2 THEN CALL CalcSmallestAmatrix(i%, a#(), dx#())
  IF predictiontype% = 1 THEN c#(i%) = BumperMajoraxis#(datasort%(i%))
  IF predictiontype% = 2 THEN c#(i%) = BumperMinorAxis#(datasort%(i%))
  IF predictiontype% = 3 THEN c#(i%) = MLIHoleDiam#(datasort%(i%))
  IF predictiontype% = 4 THEN c#(i%) = MLIMassLoss#(datasort%(i%))
  IF predictiontype% = 5 THEN c#(i%) = PressWallMajAxis#(datasort%(i%))
  IF predictiontype% = 6 THEN c#(i%) = PressWallMinAxis#(datasort%(i%))
NEXT i%
meandistance#(solution%) = meandistance#(solution%) / order%

```

The following statements are a Gauss Elimination Solver.

```

FOR i% = 1 TO order% - 1
  FOR j% = i% + 1 TO order%
    IF a#(j%, i%) <= a#(i%, i%) THEN GOTO L1
  FOR k% = i% TO order%
    dumb# = a#(i%, k%)
    a#(i%, k%) = a#(j%, k%)

```


POLYMETH.BAS

```

a#(j%, k%) = dumb#
NEXT k%
dumb# = c#(i%)
c#(i%) = c#(j%)
c#(j%) = dumb#
L1:
NEXT j%
FOR j% = i% + 1 TO order%
factor# = a#(j%, i%) / a#(i%, i%)
FOR k% = i% TO order%
a#(j%, k%) = a#(j%, k%) - a#(i%, k%) * factor#
NEXT k%
c#(j%) = c#(j%) - c#(i%) * factor#
NEXT j%
NEXT i%
FOR i% = order% TO 1 STEP -1
sum# = 0
IF i% = order% THEN
b#(i%) = (c#(i%) - sum#) / a#(i%, i%)
GOTO L2
END IF
FOR j% = i% + 1 TO order%
sum# = sum# + a#(i%, j%) * b#(j%)
NEXT j%
b#(i%) = (c#(i%) - sum#) / a#(i%, i%)
L2:
NEXT i%
,
, End of Gauss Elimination Solver.
,
prediction#(solution%) = b#(1) 'The constant coefficient is the prediction -
Here a check is made to ensure that the calculated result is reasonable.
IF predictiontype% = 1 THEN
IF prediction#(solution%) < 0# OR prediction#(solution%) > 100# * BumpMajA
END IF
IF predictiontype% = 2 THEN
IF prediction#(solution%) < 0# OR prediction#(solution%) > 100# * BumpMinA
END IF
IF predictiontype% = 3 THEN
IF prediction#(solution%) < 0# OR prediction#(solution%) > 100# * MLIHoleD
END IF
IF predictiontype% = 4 THEN
IF prediction#(solution%) < 0# OR prediction#(solution%) > 100# * MLIMassL
END IF
IF predictiontype% = 5 THEN
IF prediction#(solution%) < 0# OR prediction#(solution%) > 100# * PressWal
END IF
IF predictiontype% = 6 THEN
IF prediction#(solution%) < 0# OR prediction#(solution%) > 100# * PressWal
END IF
KEY(1) OFF
KEY(2) OFF
KEY(3) OFF
COLOR 11, 0
LOCATE 2 * solution% - 1, 1
IF predictiontype% = 1 THEN PRINT "Predicted Bumper Hole Major Axis (in) =";
IF predictiontype% = 2 THEN PRINT "Predicted Bumper Hole Minor Axis (in) =";
IF predictiontype% = 3 THEN PRINT "Predicted MLI Hole Diameter (in) =";
IF predictiontype% = 4 THEN PRINT "Predicted MLI Mass Loss (grams) =";
IF predictiontype% = 5 THEN PRINT "Pred. Press Wall Hole Major Axis (in) =";

```

POLYMETH.BAS

```

IF predictiontype% = 6 THEN PRINT "Pred. Press Wall Hole Minor Axis (in) =";
COLOR 15, 0
PRINT USING "##.###^ ^ ^ ^ ^ "; prediction#(solution%);
COLOR 14, 0
PRINT " mean data dist. = ";
COLOR 15, 0
PRINT USING "##.###^ ^ ^ ^ ^ "; meandistance#(solution%)
solution% = solution% + 1
IF solution% <= numsolutions% THEN GOTO FindSolution
CALL weightedprediction
LOCATE 13, 1
COLOR 10, 0
PRINT "Weighted Sum of Above Predictions: ";
COLOR 15, 0
PRINT USING "##.###^ ^ ^ ^ ^"; finalprediction#
LOCATE 20, 20
PRINT "
LOCATE 21, 1
PRINT "
LOCATE 22, 20
PRINT "
LOCATE 23, 20
PRINT "
LOCATE 24, 20
PRINT "
LOCATE 23, 1
COLOR 9, 0
PRINT "Do you wish to make more predictions (y/n)? ";
COLOR 12, 0
INPUT "", answer$
COLOR 15, 0
IF answer$ = "" THEN
    LOCATE 23, 1
    COLOR 9, 0
    PRINT "Do you wish to make more predictions (y/n)? ";
    COLOR 12, 0
    INPUT "", answer$
    COLOR 15, 0
END IF
answer$ = LCASE$(answer$)
COLOR 15, 0
IF answer$ = "y" THEN GOTO predictvalue
END 'End of program.
,
TestDataFileError:
COLOR 12, 0
LOCATE 4, 1
PRINT "Please Re-enter File Name (or enter QUIT to stop)"
LOCATE row%, col%
PRINT "
LOCATE row%, col%
INPUT "", MLITestDataFile$
MLITestDataFile$ = UCASE$(MLITestDataFile$)
IF MLITestDataFile$ = "QUIT" THEN END
LOCATE 4, 1
PRINT "
RESUME
,
RankError:
FOR i% = 1 TO 2 ^ numactivedof%
```

POLYMETH.BAS

```

tryagain:
  try% = numdata% * RND + 1
  IF try% > numdata% THEN try% = numdata%
  IF i% > 1 THEN
    FOR j% = 1 TO i% - 1
      IF try% = datasort%(j%) THEN GOTO tryagain
    NEXT j%
    datasort%(i%) = try%
  ELSE
    datasort%(1) = try%
  END IF

```

```

NEXT i%
RESUME CalculateCoefficients

```

```

stopcalculations:
END

```

```

simplificalculations:
amatrixcode% = 1
LOCATE 21, 1
PRINT "
LOCATE 21, 20
COLOR 10, 0
PRINT "Currently Seeking Simpler Function"
RESUME CalculateCoefficients

```

```

simplestcalculations:
amatrixcode% = 2
LOCATE 21, 1
PRINT "
LOCATE 21, 20
COLOR 10, 0
PRINT "Currently Seeking Simplest Function"
RESUME CalculateCoefficients

```

```

SUB CalcFullAmatrix (i%, a#(), dx#())
' Here the coefficients for the complete function are calculated.

```

```

a#(i%, 1) = 1#
a#(i%, 2) = dx#(1)
a#(i%, 3) = dx#(2)
a#(i%, 4) = dx#(1) * dx#(2)
a#(i%, 5) = dx#(3)
a#(i%, 6) = dx#(1) * dx#(3)
a#(i%, 7) = dx#(2) * dx#(3)
a#(i%, 8) = dx#(1) * dx#(2) * dx#(3)
a#(i%, 9) = dx#(4)
a#(i%, 10) = dx#(1) * dx#(4)
a#(i%, 11) = dx#(2) * dx#(4)
a#(i%, 12) = dx#(3) * dx#(4)
a#(i%, 13) = dx#(1) * dx#(2) * dx#(4)
a#(i%, 14) = dx#(1) * dx#(3) * dx#(4)
a#(i%, 15) = dx#(2) * dx#(3) * dx#(4)
a#(i%, 16) = dx#(1) * dx#(2) * dx#(3) * dx#(4)
END SUB

```

```

SUB CalcSmallAmatrix (i%, a#(), dx#())
' Here the coefficients for the simpler function are evaluated.

```

```

a#(i%, 1) = 1#
a#(i%, 2) = dx#(1)
a#(i%, 3) = dx#(2)

```

POLYMETH.BAS

```

a#(i%, 4) = dx#(1) * dx#(2)
a#(i%, 5) = dx#(3)
a#(i%, 6) = dx#(1) * dx#(3)
a#(i%, 7) = dx#(2) * dx#(3)
a#(i%, 8) = dx#(4)
a#(i%, 9) = dx#(1) * dx#(4)
a#(i%, 10) = dx#(2) * dx#(4)
a#(i%, 11) = dx#(3) * dx#(4)
END SUB

```

```

SUB CalcSmallestAmatrix (i%, a#(), dx#())
' Here the coefficients for the simpler function are evaluated.
a#(i%, 1) = 1#
a#(i%, 2) = dx#(1)
a#(i%, 3) = dx#(2)
a#(i%, 4) = dx#(3)
a#(i%, 5) = dx#(4)
END SUB

```

```

SUB ParamWarning
COLOR 12, 0
CLS
LOCATE 1, 1
PRINT "WARNING - This program assumes that only the following system"
PRINT "          parameters vary significantly in the database:"
PRINT
COLOR 14, 0
PRINT "          1. Bumper Thickness"
PRINT "          2. Projectile Diameter"
PRINT "          3. Projectile Velocity"
PRINT "          4. Impact Angle"
PRINT
COLOR 12, 0
PRINT "All other system parameters are assumed to be constant throughout"
PRINT "the entire database or are assumed to have no influence on the"
PRINT "amount of impact damage sustained."
PRINT
COLOR 9, 0
PRINT "Do you wish to continue (y/n)? ";
COLOR 12, 0
INPUT "", answer$
IF answer$ = "" THEN answer$ = "y"
answer$ = LCASE$(answer$)
IF answer$ = "n" THEN END
COLOR 15, 0
END SUB

```

```

SUB weightedprediction
' Here a weighted prediction is made based on all calculated results.
' Find the two predictions with largest meandistance# values
' and ignor these values when making the weighted prediction.
largemeandist1# = meandistance#(1)
numlargemeandist1% = 1
FOR i% = 2 TO numsolutions%
  IF meandistance#(i%) > largemeandist1# THEN
    largemeandist1# = meandistance#(i%)
    numlargemeandist1% = i%
  END IF
NEXT i%
IF numlargemeandist1% <> 1 THEN

```

POLYMETH.BAS

```
    largemeandist2# = meandistance#(1)
    numlargemeandist1% = 1
ELSE
    largemeandist2# = meandistance#(2)
    numlargemeandist1% = 2
END IF
FOR i% = 1 TO numsolutions%
    IF i% <> numlargemeandist1% THEN
        IF meandistance#(i%) > largemeandist2# THEN
            largemeandist2# = meandistance#(i%)
            numlargemeandist2% = i%
        END IF
    END IF
NEXT i%
factor# = 0#
finalprediction# = 0#
n% = numactivedof% - 1
IF n% < 1 THEN n% = 1
FOR i% = 1 TO numsolutions%
    IF i% <> numlargemeandist1% AND i% <> numlargemeandist2% THEN
        finalprediction# = finalprediction# + prediction#(i%) / meandistance#(i%)
        factor# = factor# + 1# / meandistance#(i%) ^ n%
    END IF
NEXT i%
finalprediction# = finalprediction# / factor#
END SUB
```

NONDIMEN.BAS

```

NONDIMEN.BAS
Source code for the nondimensional functions damage prediction program

This program was written by William K. Rule, University of Alabama, (205)3
This program makes predictions for the following functions:
1. Bumper hole minor diameter.
2. Bumper hole major diameter.
3. MLI hole diameter.
4. Pressure wall hole diameter.
This program uses functions of the form given in the report:
Schonberg, W. P., Bean, A. J., and Darzi, K., "Hypervelocity Impact Phys

DECLARE SUB OptParameters (InputIteration#, InputAlpha#)
DECLARE SUB ShowCoefficients (RSquaredValues#())
DECLARE SUB ObjectiveFunction (calc%, code%, xtry#(), objective#, rsquared#)
DECLARE SUB DisplayConvergence (col%, rsquared#)
DECLARE SUB RuleOpt (InputIteration#, InputAlpha#, RSquaredValues#())
DECLARE SUB MakePrediction ()

COMMON SHARED PredictBumpThick#, PredictBumpStandOff#, PredictPressWallThick#, P
COMMON SHARED pi#, a#(), BumperSoundSpeed#, numdata%
COMMON SHARED BumpMinDiaPred#, BumpMaxDiaPred#, MLIDiaPred#, PressureWallDiaPred
COMMON SHARED BumperThickness#(), BumperStandOff#(), PressureWallThickness#(), P
COMMON SHARED BumperMajorAxis#(), BumperMinorAxis#(), MLIHoleDiam#(), MLIMassLos

' Vector a#() stores the function coefficients.
DIM a#(1 TO 23)
' Dimensioned for 100 data points.
DIM TestID$(1 TO 100)
DIM DataSource$(1 TO 100)
DIM TestDate$(1 TO 100)
DIM BumperThickness#(1 TO 100)
DIM BumperStandOff#(1 TO 100)
DIM PressureWallThickness#(1 TO 100)
DIM ProjectileDiameter#(1 TO 100)
DIM ImpactAngle#(1 TO 100)
DIM ProjectileVelocity#(1 TO 100)
DIM BumperMajorAxis#(1 TO 100)
DIM BumperMinorAxis#(1 TO 100)
DIM MLIHoleDiam#(1 TO 100)
DIM MLIMassLoss#(1 TO 100)
DIM PressWallMajAxis#(1 TO 100)
DIM PressWallMinAxis#(1 TO 100)
' Vector RSquaredValues#() stores the coefficients of determination for the
DIM RSquaredValues#(4)

pi# = 3.14159265359#

COLOR 9, 0
CLS
LOCATE 1, 1
PRINT CHR$(201);
FOR i% = 1 TO 78
PRINT CHR$(205);
NEXT i%
PRINT CHR$(187)
LOCATE 2, 1: PRINT CHR$(186)

```

CONDIMEN.BAS

```
LOCATE 2, 80: PRINT CHR$(186)
LOCATE 3, 1
PRINT CHR$(200);
FOR i% = 1 TO 78
  PRINT CHR$(205);
NEXT i%
PRINT CHR$(188)
```

TestDataFileError is used to trap user input file name errors.

```
ON ERROR GOTO TestDataFileError
```

```
LOCATE 2, 3
```

```
row% = 2
```

```
col% = 24
```

```
COLOR 11, 0
```

```
PRINT "Test Data File Name? ";
```

```
COLOR 12, 0
```

MLITestDataFile\$ contains the test data in a format compatible with that of

```
INPUT "", MLITestDataFile$
```

```
OPEN MLITestDataFile$ FOR INPUT AS #1
```

```
ON ERROR GOTO 0
```

```
COLOR 9, 0
```

```
LOCATE 3, 1
```

```
PRINT CHR$(204)
```

```
LOCATE 3, 40
```

```
PRINT CHR$(203)
```

```
LOCATE 3, 80
```

```
PRINT CHR$(185)
```

```
LOCATE 4, 1
```

```
PRINT CHR$(186)
```

```
LOCATE 4, 40
```

```
PRINT CHR$(186)
```

```
LOCATE 4, 80
```

```
PRINT CHR$(186)
```

```
LOCATE 5, 1
```

```
PRINT CHR$(200)
```

```
LOCATE 5, 40
```

```
PRINT CHR$(202)
```

```
LOCATE 5, 80
```

```
PRINT CHR$(188)
```

```
FOR i% = 2 TO 79
```

```
  LOCATE 5, i%
```

```
  IF i% <> 40 THEN PRINT CHR$(205)
```

```
NEXT i%
```

```
InputBumperElasticModulus:
```

```
LOCATE 4, 2
```

```
COLOR 11, 0
```

```
PRINT "Bumper Elastic Modulus (MPa)? ";
```

```
COLOR 12, 0
```

```
INPUT "", BumperElasticModulus#
```

```
IF BumperElasticModulus# <= 0# THEN
```

```
  LOCATE 6, 1
```

```
  COLOR 11, 9
```

```
  PRINT "Sorry - Bumper Elastic Modulus Must Be > Zero!"
```

```
  GOTO InputBumperElasticModulus
```

```
END IF
```

```
InputBumperMassDensity:
```

```
LOCATE 4, 42
```

NONDIMEN.BAS

```

COLOR 11, 0
PRINT " Bumper Mass Density (kg/m^3)? ";
COLOR 12, 0
INPUT "", BumperMassDensity#
IF BumperMassDensity# <= 0# THEN
    LOCATE 6, 1
    COLOR 11, 9
    PRINT "Sorry - Bumper Mass Density Must Be > Zero!  "
    GOTO InputBumperMassDensity
END IF
BumperSoundSpeed# = SQR(BumperElasticModulus# / BumperMassDensity#)
'
'    numdata% stores the number of data records in the database.
numdata% = 0
'    The following variables store averages of the database records.
BumpThkAve# = 0
BumpStandOffAve# = 0
PressWallThkAve# = 0
ProjDiaAve# = 0
ImpAngAve# = 0
ProjVelAve# = 0
BumpMajAxisAve# = 0
BumpMinAxisAve# = 0
MLIHoleDiamAve# = 0
MLIMassLossAve# = 0
PressWallMajAxisAve# = 0
PressWallMinAxisAve# = 0
'
'Modify top line of boarder.
COLOR 9, 0
LOCATE 5, 1
PRINT CHR$(204)
LOCATE 5, 80
PRINT CHR$(185)
'
'    Scroll through data.
VIEW PRINT 6 TO 12
'
DO WHILE NOT EOF(1)
    numdata% = numdata% + 1
    INPUT #1, Dummy$
    INPUT #1, TestID$(numdata%)
    INPUT #1, DataSource$(numdata%)
    INPUT #1, TestDate$(numdata%)
    COLOR 9, 0
    PRINT CHR$(186);
    COLOR 11, 0
    PRINT " No.: ";
    PRINT numdata%;
    PRINT "   ID: ";
    PRINT TestID$(numdata%);
    PRINT "   Source: ";
    PRINT DataSource$(numdata%);
    PRINT "   Date: ";
    PRINT TestDate$(numdata%);
    LOCATE CSRLIN, 80
    COLOR 9, 0
    PRINT CHR$(186)
    INPUT #1, Dummy# 'Skip bumper material field.
    INPUT #1, BumperThickness$(numdata%)

```


NONDIMEN.BAS

```

BumpThkAve# = BumpThkAve# + BumperThickness#(numdata%)
INPUT #1, BumperStandOff#(numdata%)
BumpStandOffAve# = BumpStandOffAve# + BumperStandOff#(numdata%)
INPUT #1, Dummy# 'Skip pressure wall material field.
INPUT #1, PressureWallThickness#(numdata%)
PressWallThkAve# = PressWallThkAve# + PressureWallThickness#(numdata%)
INPUT #1, Dummy# 'Skip projectile material field.
INPUT #1, ProjectileDiameter#(numdata%)
ProjDiaAve# = ProjDiaAve# + ProjectileDiameter#(numdata%)
INPUT #1, ImpactAngle#(numdata%)
ImpAngAve# = ImpAngAve# + ImpactAngle#(numdata%)
INPUT #1, ProjectileVelocity#(numdata%)
ProjVelAve# = ProjVelAve# + ProjectileVelocity#(numdata%)
INPUT #1, BumperMajorAxis#(numdata%)
BumpMajAxisAve# = BumpMajAxisAve# + BumperMajorAxis#(numdata%)
INPUT #1, BumperMinorAxis#(numdata%)
BumpMinAxisAve# = BumpMinAxisAve# + BumperMinorAxis#(numdata%)
INPUT #1, MLIHoleDiam#(numdata%)
MLIHoleDiamAve# = MLIHoleDiamAve# + MLIHoleDiam#(numdata%)
INPUT #1, MLIMassLoss#(numdata%)
MLIMassLossAve# = MLIMassLossAve# + MLIMassLoss#(numdata%)
INPUT #1, PressWallMajAxis#(numdata%)
PressWallMajAxisAve# = PressWallMajAxisAve# + PressWallMajAxis#(numdata%)
INPUT #1, PressWallMinAxis#(numdata%)
PressWallMinAxisAve# = PressWallMinAxisAve# + PressWallMinAxis#(numdata%)
INPUT #1, Dummy$
LOOP
'
VIEW PRINT
'
' Build box for averages.
LOCATE 12, 2
FOR i% = 1 TO 78
PRINT CHR$(205);
NEXT i%
LOCATE 12, 1: PRINT CHR$(204)
LOCATE 12, 40: PRINT CHR$(203)
LOCATE 12, 80: PRINT CHR$(185)
FOR i% = 13 TO 23
LOCATE i%, 1: PRINT CHR$(186)
LOCATE i%, 40: PRINT CHR$(186)
LOCATE i%, 80: PRINT CHR$(186)
NEXT i%
LOCATE 24, 1: PRINT CHR$(200);
FOR i% = 1 TO 78
IF i% <> 39 THEN PRINT CHR$(205);
IF i% = 39 THEN PRINT CHR$(202);
NEXT i%
PRINT CHR$(188)
' Calculate and print out parameter averages.
BumpThkAve# = BumpThkAve# / numdata%
LOCATE 13, 3
COLOR 11, 0
PRINT "Ave. Bumper Thk (in):";
COLOR 15, 0
PRINT USING "##.###^"; BumpThkAve#;
BumpStandOffAve# = BumpStandOffAve# / numdata%
LOCATE 13, 42
COLOR 11, 0
PRINT "Ave. Bump. Stand Off (in):";

```

NONDIMEN.BAS

```

COLOR 15, 0
PRINT USING "##.###^"; BumpStandOffAve#;
PressWallThkAve# = PressWallThkAve# / numdata%
LOCATE 15, 3
COLOR 11, 0
PRINT "Ave. Pres Wall Thk (in):";
COLOR 15, 0
PRINT USING "##.###^"; PressWallThkAve#;
ProjDiaAve# = ProjDiaAve# / numdata%
LOCATE 15, 42
COLOR 11, 0
PRINT "Ave. Proj. Dia. (in):";
COLOR 15, 0
PRINT USING "##.###^"; ProjDiaAve#;
ImpAngAve# = ImpAngAve# / numdata%
LOCATE 17, 3
COLOR 11, 0
PRINT "Ave. Impact Angle (deg):";
COLOR 15, 0
PRINT USING "##.###^"; ImpAngAve#;
ProjVelAve# = ProjVelAve# / numdata%
LOCATE 17, 42
COLOR 11, 0
PRINT "Ave. Proj. Vel. (km/sec):";
COLOR 15, 0
PRINT USING "##.###^"; ProjVelAve#;
BumpMajAxisAve# = BumpMajAxisAve# / numdata%
LOCATE 19, 3
COLOR 11, 0
PRINT "Ave. Maj. Bumper Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; BumpMajAxisAve#;
BumpMinAxisAve# = BumpMinAxisAve# / numdata%
LOCATE 19, 42
COLOR 11, 0
PRINT "Ave. Min. Bumper Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; BumpMinAxisAve#;
MLIHoleDiamAve# = MLIHoleDiamAve# / numdata%
LOCATE 21, 3
COLOR 11, 0
PRINT "Ave. MLI Hole Diam. (in):";
COLOR 15, 0
PRINT USING "##.###^"; MLIHoleDiamAve#;
MLIMassLossAve# = MLIMassLossAve# / numdata%
LOCATE 21, 42
COLOR 11, 0
PRINT "Ave. MLI Mass Loss (grams):";
COLOR 15, 0
PRINT USING "##.###^"; MLIMassLossAve#;
PressWallMajAxisAve# = PressWallMajAxisAve# / numdata%
LOCATE 23, 3
COLOR 11, 0
PRINT "Ave. Maj. P.Wall Hole (in):";
COLOR 15, 0
PRINT USING "##.###^"; PressWallMajAxisAve#;
PressWallMinAxisAve# = PressWallMinAxisAve# / numdata%
LOCATE 23, 42
COLOR 11, 0
PRINT "Ave. Min. P.Wall Hole (in):";

```

NDIMEN.BAS

```
DLOR 15, 0
PRINT USING "##.###^"; PressWallMinAxisAve#;
LOCATE 25, 28
DLOR 12, 0
PRINT "press any key to continue";
)
DO WHILE INKEY$ = "" 'Press any key to continue
```

RuleOpt calculates the function coefficients and stores them in vector a#(1 t
Prompt user for optimizer parameters.

```
ALL OptParameters(InputIteration#, InputAlpha#)
RuleOpt uses a modified Powell's method (as developed by W.K.Rule) to optimal
ALL RuleOpt(InputIteration#, InputAlpha#, RSquaredValues#())
```

ShowCoefficients displays the calculated coefficients.
ALL ShowCoefficients(RSquaredValues#())

Request the user for parameter values to be used for damage predictions.
redictValue:

```
DLOR 11, 0
LS
LOCATE 1, 1
PRINT "ENTER DATA FOR DESIRED PREDICTION: "
```

InputBumperThickness:

```
LOCATE 5, 1
DLOR 11, 0
PRINT "Bumper Thickness (in): ";
DLOR 12, 0
INPUT "", PredictBumpThick#
IF PredictBumpThick# <= 0# THEN
  LOCATE 6, 1
  COLOR 11, 9
  PRINT "Sorry - Bumper Thickness Must Be > Zero!"
  GOTO InputBumperThickness
ND IF
LOCATE 6, 1: PRINT "
```

InputBumperStandOff:

```
LOCATE 7, 1
DLOR 11, 0
PRINT "Bumper Stand-Off (in): ";
DLOR 12, 0
INPUT "", PredictBumpStandOff#
IF PredictBumpStandOff# <= 0# THEN
  LOCATE 8, 1
  COLOR 11, 9
  PRINT "Sorry - Bumper Stand-Off Must Be > Zero!"
  GOTO InputBumperStandOff
ND IF
LOCATE 8, 1: PRINT "
```

InputPressureWallThickness:

```
LOCATE 9, 1
DLOR 11, 0
PRINT "Pressure Wall Thickness (in): ";
DLOR 12, 0
INPUT "", PredictPressWallThick#
IF PredictPressWallThick# <= 0# THEN
  LOCATE 10, 1
```

NONDIMEN.BAS

```

    COLOR 11, 9
    PRINT "Sorry - Pressure Wall Thickness Must Be > Zero!"
    GOTO InputPressureWallThickness
END IF
LOCATE 10, 1: PRINT "
,
InputProjectileDiameter:
LOCATE 11, 1
COLOR 11, 0
PRINT "Projectile Diameter (in):      ";
COLOR 12, 0
INPUT "", PredictProjDia#
IF PredictProjDia# <= 0# THEN
    LOCATE 12, 1
    COLOR 11, 9
    PRINT "Sorry - Projectile Diameter Must Be > Zero!"
    GOTO InputProjectileDiameter
END IF
LOCATE 12, 1: PRINT "
,
InputImpactAngle:
LOCATE 13, 1
COLOR 11, 0
PRINT "Impact Angle (degrees):      ";
COLOR 12, 0
INPUT "", PredictImpAngle#
IF PredictImpAngle# < 0# THEN
    LOCATE 14, 1
    COLOR 11, 9
    PRINT "Sorry - Impact Angle Must Be >= Zero!"
    GOTO InputImpactAngle
END IF
LOCATE 14, 1: PRINT "
,
InputProjectileVelocity:
LOCATE 15, 1
COLOR 11, 0
PRINT "Projectile Velocity (km/sec): ";
COLOR 12, 0
INPUT "", PredictProjVel#
IF PredictProjVel# <= 0# THEN
    LOCATE 16, 1
    COLOR 11, 9
    PRINT "Sorry - Projectile Velocity Must Be > Zero!"
    GOTO InputProjectileVelocity
END IF
LOCATE 16, 1: PRINT "
,
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = ""           'Press any key to continue
,
' MakePrediction evaluates the damage functions at user input values.
CALL MakePrediction
,
' Show predictions on the screen, and associated function R-squared values (coe
CLS
LOCATE 1, 1

```

NDIMEN.BAS

```

OLOR 11, 0
RINT "Calculated Results:      ";
OLOR 10, 0
RINT "(Function R-Squared Values Given In Brackets)"

```

If the impact angle for the prediction is zero, then set the minimum and maximum bumper hole diameters to be equal to their average.

```

F PredictImpAngle# = 0# THEN
  AverageDiameter# = (BumpMaxDiaPred# + BumpMinDiaPred#) / 2#
  BumpMinDiaPred# = AverageDiameter#
  BumpMaxDiaPred# = AverageDiameter#

```

```

ND IF
OCATE 5, 1
OLOR 12, 0
RINT "Minimum Bumper Hole Diameter (in): ";
OLOR 15, 0
RINT USING "####.#### "; BumpMinDiaPred#;
OLOR 10, 0
RINT USING "      (##.####) "; RSquaredValues#(1)

```

```

OCATE 7, 1
OLOR 12, 0
RINT "Maximum Bumper Hole Diameter (in): ";
OLOR 15, 0
RINT USING "####.#### "; BumpMaxDiaPred#;
OLOR 10, 0
RINT USING "      (##.####) "; RSquaredValues#(2)

```

```

OCATE 9, 1
OLOR 12, 0
RINT "MLI Hole Diameter (in):           ";
OLOR 15, 0
RINT USING "####.#### "; MLIDiaPred#;
OLOR 10, 0
RINT USING "      (##.####) "; RSquaredValues#(3)

```

```

OCATE 11, 1
OLOR 12, 0
RINT "Pressure Wall Hole Diameter (in): ";
OLOR 15, 0
RINT USING "####.#### "; PressureWallDiaPred#;
OLOR 10, 0
RINT USING "      (##.####) "; RSquaredValues#(4)

```

Allow the user to make multiple predictions from the same set of coefficients

```

OCATE 24, 1
OLOR 11, 0
RINT "Do You Wish To Enter Data For Another Prediction (y/n)? ";
OLOR 12, 0
INPUT "", Answer$
OLOR 15, 0
F Answer$ = "" THEN Answer$ = "y"
Answer$ = LCASE$(Answer$)
F Answer$ = "y" THEN GOTO PredictValue
ND

```

This subroutine traps database input file errors.

```

TestDataFileError:
OLOR 12, 0
OCATE 4, 1
RINT "Please Re-enter File Name (or enter QUIT to stop)"

```

NONDIMEN.BAS

```

LOCATE row%, col%
PRINT " "
LOCATE row%, col%
INPUT "", MLITestDataFile$
MLITestDataFile$ = UCASE$(MLITestDataFile$)
IF MLITestDataFile$ = "QUIT" THEN END
LOCATE 4, 1
PRINT " "
RESUME
QuitRunning:
END
RETURN

SUB DisplayConvergence (col%, rsquared#)
' This subroutine displays the effectiveness of the coefficient optimizer fo
' This subroutine is designed to fill the screen with data, rather than scro
' Only rows 3 thru 23 are used to display the data.
CurRow% = CSRLIN
CurCol% = POS(0)
IF CurRow% = 24 THEN
CurRow% = 5
col% = col% + 10
IF col% > 80 THEN
col% = 1
COLOR 15, 0
Here we clear the screen of data if it is full.
FOR clrline% = 5 TO 23
LOCATE clrline%, 1: PRINT "
NEXT clrline%
COLOR 15, 9
END IF
END IF
' rsquared# is the current value of the coefficient of determination (R^2) o
LOCATE CurRow%, col%
PRINT USING " #.#### "; rsquared#
END SUB

SUB MakePrediction
' This subroutine uses the calculated function coefficients a#() to make pre
'
' Convert impact angle to radians.
angle# = PredictImpAngle# * pi# / 180#
'
' BumpMinDiaPred# is the predicted value of the bumper minimum hole diameter
BumpMinDiaPred# = a#(1) * (PredictProjVel# / BumperSoundSpeed#) ^ a#(2)
BumpMinDiaPred# = BumpMinDiaPred# * (PredictBumpThick# / PredictProjDia#) ^ a#(3)
BumpMinDiaPred# = BumpMinDiaPred# * (COS(angle#)) ^ a#(4) + a#(5)
BumpMinDiaPred# = BumpMinDiaPred# * PredictProjDia#
IF BumpMinDiaPred# < 0# THEN BumpMinDiaPred# = 0#
'
' BumpMaxDiaPred# is the predicted value of the bumper maximum hole diameter
BumpMaxDiaPred# = a#(6) * (PredictProjVel# / BumperSoundSpeed#) ^ a#(7)
BumpMaxDiaPred# = BumpMaxDiaPred# * (PredictBumpThick# / PredictProjDia#) ^ a#(8)
BumpMaxDiaPred# = BumpMaxDiaPred# * (COS(angle#)) ^ a#(9) + a#(10)
BumpMaxDiaPred# = BumpMaxDiaPred# * PredictProjDia#
IF BumpMaxDiaPred# < 0# THEN BumpMaxDiaPred# = 0#
'
' MLIDiaPred# is the predicted value of the MLI hole diameter.
MLIDiaPred# = a#(11) * (PredictProjVel# / BumperSoundSpeed#) ^ a#(12)
MLIDiaPred# = MLIDiaPred# * (PredictBumpThick# / PredictProjDia#) ^ a#(13)

```

NDIMEN.BAS

```
IDiaPred# = MLIDiaPred# * (PredictBumpStandOff# / PredictProjDia#) ^ a#(14)
IDiaPred# = MLIDiaPred# * (COS(angle#)) ^ a#(15) + a#(16)
IDiaPred# = MLIDiaPred# * PredictProjDia#
MLIDiaPred# < 0# THEN MLIDiaPred# = 0#
```

PressureWallDiaPred# is the predicted value of the average pressure wall h

```
essureWallDiaPred# = a#(17) * (PredictProjVel# / BumperSoundSpeed#) ^ a#(18)
essureWallDiaPred# = PressureWallDiaPred# * (PredictBumpThick# / PredictProjDi
essureWallDiaPred# = PressureWallDiaPred# * (PredictBumpStandOff# / PredictPro
essureWallDiaPred# = PressureWallDiaPred# * (PredictPressWallThick# / PredictP
essureWallDiaPred# = PressureWallDiaPred# * (COS(angle#)) ^ a#(22) + a#(23)
essureWallDiaPred# = PressureWallDiaPred# * PredictProjDia#
PressureWallDiaPred# < 0# THEN PressureWallDiaPred# = 0#
```

D SUB

```
ObjectiveFunction (calc%, code%, xtry#(), objective#, rsquared#)
  This subroutine calculates the objective function for the optimizer.
  Here the objective function is the coefficient of determination (R^2) of t
  objective# is the objective function.
  xtry#() is a vector of trial coefficient values used by the optimizer.
  calc% = 0 means do not calculate R^2, calc% = 1 means do calculate R^2.
  objective# = 0#
  avemeasured# is the average measured value of the dependent parameter (use
  vemeasured# = 0#
  numdata% is the total number of records in the database.
  OR datacount% = 1 TO numdata%
  Convert impact angle to radians.
  angle# = ImpactAngle#(datacount%) * pi# / 180#
  code% equal to 1 means treat bumper hole minor diameter function.
  IF code% = 1 THEN
    measured# is the measured value of the dependent variable.
    measured# = BumperMinorAxis#(datacount%) / ProjectileDiameter#(datacount%)
    calculated# is the calculated value of the dependent variable.
    calculated# = xtry#(1) * (ProjectileVelocity#(datacount%) / BumperSoundSpe
    calculated# = calculated# * (BumperThickness#(datacount%) / ProjectileDiam
    calculated# = calculated# * (COS(angle#)) ^ xtry#(4) + xtry#(5)
  END IF
  code% equal to 2 means treat bumper hole major diameter function.
  IF code% = 2 THEN
    measured# = BumperMajorAxis#(datacount%) / ProjectileDiameter#(datacount%)
    calculated# = xtry#(1) * (ProjectileVelocity#(datacount%) / BumperSoundSpe
    calculated# = calculated# * (BumperThickness#(datacount%) / ProjectileDiam
    calculated# = calculated# * (COS(angle#)) ^ xtry#(4) + xtry#(5)
  END IF
  code% equal to 3 means treat MLI hole diameter function.
  IF code% = 3 THEN
    measured# = MLIHoleDiam#(datacount%) / ProjectileDiameter#(datacount%)
    calculated# = xtry#(1) * (ProjectileVelocity#(datacount%) / BumperSoundSpe
    calculated# = calculated# * (BumperThickness#(datacount%) / ProjectileDiam
    calculated# = calculated# * (BumperStandOff#(datacount%) / ProjectileDiam
    calculated# = calculated# * (COS(angle#)) ^ xtry#(5) + xtry#(6)
  END IF
  code% equal to 4 means treat pressure wall average hole diameter function.
  IF code% = 4 THEN
    AverageDiameter# = (PressWallMinAxis#(datacount%) + PressWallMajAxis#(data
    measured# = AverageDiameter# / ProjectileDiameter#(datacount%)
    calculated# = xtry#(1) * (ProjectileVelocity#(datacount%) / BumperSoundSpe
    calculated# = calculated# * (BumperThickness#(datacount%) / ProjectileDiam
    calculated# = calculated# * (BumperStandOff#(datacount%) / ProjectileDiam
```

NONDIMEN.BAS

```

        calculated# = calculated# * (PressureWallThickness#(datacount%) / Project
        calculated# = calculated# * (COS(angle#)) ^ xtry#(6) + xtry#(7)
    END IF
    objective# = objective# + (measured# - calculated#) ^ 2
    avemeasured# = avemeasured# + measured#
NEXT datacount%
'
IF calc% = 1 THEN
    avemeasured# = avemeasured# / numdata%
    value# = 0#
    '       Here value# is determined which is used in the calculation of R^2.
    FOR datacount% = 1 TO numdata%
        IF code% = 1 THEN
            measured# = BumperMinorAxis#(datacount%) / ProjectileDiameter#(datacount%)
        END IF
        IF code% = 2 THEN
            measured# = BumperMajorAxis#(datacount%) / ProjectileDiameter#(datacount%)
        END IF
        IF code% = 3 THEN
            measured# = MLIHoleDiam#(datacount%) / ProjectileDiameter#(datacount%)
        END IF
        IF code% = 4 THEN
            AverageDiameter# = (PressWallMinAxis#(datacount%) + PressWallMajAxis#(datacount%)) / 2
            measured# = AverageDiameter# / ProjectileDiameter#(datacount%)
        END IF
        value# = value# + (measured# - avemeasured#) ^ 2
    NEXT datacount%
    rsquared# = 1# - objective# / value#
END IF
END SUB

SUB OptParameters (InputIteration#, InputAlpha#)
    COLOR 11, 0
    CLS
    LOCATE 1, 1
    PRINT "Enter the ";
    COLOR 14, 1
    PRINT "ITERATION PARAMETER";
    COLOR 11, 0
    PRINT " for the function coefficient optimizer."
    PRINT "Values in the range ";
    COLOR 14, 1
    PRINT " (10 to 1000) ";
    COLOR 11, 0
    PRINT " are acceptable, ";
    COLOR 14, 1
    PRINT " 20 is recommended."
    COLOR 11, 0
    PRINT "High values will tend to produce better results but longer execution"
    PRINT "times."
    PRINT
    PRINT
    COLOR 14, 1
    INPUT " ITERATION PARAMETER? ", InputIteration#
    '
    COLOR 11, 0
    LOCATE 10, 1
    PRINT "Enter the ";
    COLOR 14, 1
    PRINT "SEARCH DOMAIN PARAMETER ";

```


NONDIMEN.BAS

```

COLOR 11, 0
PRINT "for the function coefficient optimizer."
PRINT "Values in the range ";
COLOR 14, 1
PRINT " (0.1 to 3) ";
COLOR 11, 0
PRINT " are acceptable, ";
COLOR 14, 1
PRINT " 1 is recommended."
COLOR 11, 0
PRINT "High values will tend to reduce the chance of getting trapped in a"
PRINT "local minimum (rather than the global minimum) but will tend to reduce"
PRINT "the chance of precisely locating the global minimum."
PRINT
PRINT
COLOR 14, 1
INPUT " SEARCH DOMAIN PARAMETER? ", InputAlpha#
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = ""           'Press any key to continue
END SUB

SUB RuleOpt (InputIteration#, InputAlpha#, RSquaredValues#())
'   This subroutine finds optimal values for the prediction function coefficient
'   Optimal in the sense that function R^2 values are minimized (nonlinear least squares)
'   The optimization technique is based on a modified Powell's method and is
'   described in the following report:
'       Rule, W.K., "ROCOPT - A User Friendly Interactive Code to Optimize
'       Rocket Structural Components," NASA CR-183837, 1989.
'
DIM search#(1 TO 7, 1 TO 7) 'Matrix of columns which are search vectors
DIM searchnew#(1 TO 7) 'New search vector generated as a vector sum of previous
DIM alpha#(1 TO 7) 'Search vector multiplier
DIM x#(1 TO 7) 'Design variables
DIM xtry#(1 TO 7) 'Trial values of design variables to check if objective function
seed% = ((TIMER * 65536) / 86400) - 32768 'Seed% is the seed number of the random number generator
RANDOMIZE seed%
ON KEY(1) GOSUB QuitRunning
KEY(1) ON
'   If code%=1 then find coefficients for BumpMinDiaPred# function.
'   If code%=2 then find coefficients for BumpMaxDiaPred# function.
'   If code%=3 then find coefficients for MLIDiaPred# function.
'   If code%=4 then find coefficients for PressureWallDiaPred# function.
code% = 1
'
FindCoefficients:
searchsequence% = 1 'This keeps track of the number of searches run for a given search sequence
Findcoefficients1:
iteration# = InputIteration#
alphatry# = InputAlpha#
COLOR 15, 0
CLS
COLOR 11, 1
IF code% = 1 THEN PRINT "Coefficient of Determination for Bumper Hole Minimum Diameter"
IF code% = 2 THEN PRINT "Coefficient of Determination for Bumper Hole Maximum Diameter"
IF code% = 3 THEN PRINT "Coefficient of Determination for MLI Hole Diameter Function"
IF code% = 4 THEN PRINT "Coefficient of Determination for Pressure Wall Hole Diameter"
LOCATE 3, 1: COLOR 0, 15

```

NONDIMEN.BAS

```

PRINT "CONDUCTING FUNCTION COEFFICIENT SEARCH SEQUENCE: ";
COLOR 14, 0
PRINT USING " # "; searchsequence%;
COLOR 0, 15
PRINT " ";
COLOR 12, 0
PRINT " PRESS F1 TO QUIT ";
COLOR 0, 15
PRINT " ";
COLOR 15, 0
PRINT
col% = 1 'This variable keeps track of the column position.
COLOR 15, 9
'
IF code% = 1 THEN numvar% = 5
IF code% = 2 THEN numvar% = 5
IF code% = 3 THEN numvar% = 6
IF code% = 4 THEN numvar% = 7
totals% = iteration# * numvar% 'totals% is the total number of search matrices t
alphamult# = .01# ^ (1# / totals%) 'This factor is to reduce alphastry#
' to 1/100 of it's initial value by the end of the iterations.
' Initialize variables.
IF searchsequence% = 1 THEN
  FOR i% = 1 TO numvar%
    x#(i%) = 0#
  NEXT i%
END IF
num% = 0 'Counter for number of search matrices generated
iteration% = 0 'Counter for number of search vectors used
start:
FOR i% = 1 TO numvar% 'Generate the random search matrix
  FOR j% = 1 TO numvar%
    search#(i%, j%) = -1# + 2# * RND
  NEXT j%
NEXT i%
FOR i% = 1 TO numvar% 'Normalize random search vectors to +/-1.
  smax# = ABS(search#(1, i%))
  FOR j% = 2 TO numvar%
    IF ABS(search#(j%, i%)) > smax# THEN smax# = ABS(search#(j%, i%))
  NEXT j%
  FOR j% = 1 TO numvar%
    search#(j%, i%) = search#(j%, i%) / smax#
  NEXT j%
NEXT i%
nexts:
calc% = 0 'calc% = 0 means do not calculate R^2, calc% = 1 means do calculate R^
FOR i% = 1 TO numvar%
  iteration% = iteration% + 1
  FOR j% = 1 TO numvar% 'Check objective function a negative distance along th
    xtry#(j%) = x#(j%) - alphastry# * search#(j%, i%)
  NEXT j%
  CALL ObjectiveFunction(calc%, code%, xtry#(), objective#, rsquared#)
  Obackward# = objective#
  FOR j% = 1 TO numvar%
    xtry#(j%) = x#(j%)
  NEXT j%
  CALL ObjectiveFunction(calc%, code%, xtry#(), objective#, rsquared#)
  Ocurrent# = objective#
  FOR j% = 1 TO numvar% 'Check objective function a positive distance along sea
    xtry#(j%) = x#(j%) + alphastry# * search#(j%, i%)

```

NONDIMEN.BAS

```

NEXT j%
CALL ObjectiveFunction(calc%, code%, xtry#(), objective#, rsquared#)
Oforward# = objective#
IF Ocurrent# < Obackward# AND Ocurrent# < Oforward# THEN
    alpha#(i%) = 0# 'Make no change if current position is better.
    GOTO nexts2
END IF
IF Obackward# >= Oforward# THEN
    alpha#(i%) = alphastry#
ELSE
    alpha#(i%) = alphastry# * (-1)
END IF
FOR j% = 1 TO numvar%
    x#(j%) = x#(j%) + alpha#(i%) * search#(j%, i%)
NEXT j%
nexts2:
    searchnew#(i%) = 0#
NEXT i%
IF nums% >= totals% THEN GOTO finish
nums% = nums% + 1
alphastry# = alphastry# * alphamult#
FOR j% = 1 TO numvar%
    xtry#(j%) = x#(j%)
NEXT j%
calc% = 1
CALL ObjectiveFunction(calc%, code%, xtry#(), objective#, rsquared#)
CALL DisplayConvergence(col%, rsquared#)
smax# = 0#
FOR i% = 1 TO numvar% 'Generate the new search vector
    FOR j% = 1 TO numvar%
        searchnew#(i%) = searchnew#(i%) + search#(i%, j%) * alpha#(j%)
    NEXT j%
    IF ABS(searchnew#(i%)) > smax# THEN smax# = ABS(searchnew#(i%))
NEXT i%
test1# = (nums% * 1#) / (numvar% * 1#)
test2# = INT((nums% * 1#) / (numvar% * 1#))
test3# = test1# - test2#
IF test3# = 0 THEN
    GOTO start
ELSE
    IF smax# = 0# THEN 'Regenerate search matrix if current one does no good
        GOTO start
    END IF
    FOR j% = 1 TO numvar%
        searchnew#(j%) = searchnew#(j%) / smax#
    NEXT j%
    FOR i% = 1 TO numvar%
        FOR j% = 1 TO numvar%
            IF i% < numvar% THEN search#(j%, i%) = search#(j%, i% + 1)
            IF i% = numvar% THEN search#(j%, i%) = searchnew#(j%)
        NEXT j%
    NEXT i%
    GOTO nexts
END IF
finish:
IF code% = 1 THEN j% = 1
IF code% = 2 THEN j% = 6
IF code% = 3 THEN j% = 11
IF code% = 4 THEN j% = 17
FOR i% = 1 TO numvar%

```

NONDIMEN.BAS

```

    a#(j% + i% - 1) = x#(i%)
NEXT i%
searchsequence% = searchsequence% + 1
IF searchsequence% <= 3 THEN GOTO Findcoefficients1
calc% = 1
CALL ObjectiveFunction(calc%, code%, x#(), objective#, rsquared#)
'    RsquaredValues#() stores the R^2 values for each prediction function.
RSquaredValues#(code%) = rsquared#
code% = code% + 1
IF code% <= 4 THEN GOTO FindCoefficients
KEY(1) OFF
END SUB

SUB ShowCoefficients (RSquaredValues#())
COLOR 11, 0
CLS
LOCATE 1, 24
PRINT "Calculated Function Coefficients"
'
COLOR 12, 0
LOCATE 3, 1
PRINT "Minimum Bumper Hole Diameter Coefficients: (R^2 = ";
PRINT USING "##.### "; RSquaredValues#(1);
PRINT ")"
LOCATE 5, 1
COLOR 15, 0
PRINT USING " ###.##### "; a#(1); a#(2); a#(3); a#(4); a#(5);
'
COLOR 13, 0
LOCATE 9, 1
PRINT "Maximum Bumper Hole Diameter Coefficients: (R^2 = ";
PRINT USING "##.### "; RSquaredValues#(2);
PRINT ")"
LOCATE 11, 1
COLOR 15, 0
PRINT USING " ###.##### "; a#(6); a#(7); a#(8); a#(9); a#(10);
'
COLOR 14, 0
LOCATE 15, 1
PRINT "MLI Hole Diameter Coefficients: (R^2 = ";
PRINT USING "##.### "; RSquaredValues#(3);
PRINT ")"
LOCATE 17, 1
COLOR 15, 0
PRINT USING " ###.##### "; a#(11); a#(12); a#(13); a#(14); a#(15); a#(16)
'
COLOR 10, 0
LOCATE 21, 1
PRINT "Pressure Wall Hole Average Diameter Coefficients: (R^2 = ";
PRINT USING "##.### "; RSquaredValues#(4);
PRINT ")"
LOCATE 23, 1
COLOR 15, 0
PRINT USING " ###.##### "; a#(17); a#(18); a#(19); a#(20); a#(21); a#(22); a#(23)
'
LOCATE 25, 28
COLOR 12, 0
PRINT "press any key to continue";
DO
LOOP WHILE INKEY$ = ""
'Press any key to continue

```

NONDIMEN.BAS

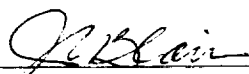
END SUB

APPROVAL

MLIBLAST - A PROGRAM TO EMPIRICALLY
PREDICT HYPERVELOCITY IMPACT DAMAGE
TO THE SPACE STATION

By William K. Rule

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.



James C. Blair
Director, Structures & Dynamics Laboratory

