Damage Resistance and Damage Tolerance of Thin Composite Facesheet Honeycomb Panels

by

Simon Charles Lie

S.B., Massachusetts Institute of Technology
(1987)

Submitted in partial fulfillment
of the requirements for the
degree of

MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

March 1989

Signature of Author _____
Department of Aeronautics and Astronautics
March 2, 1989

Certified by _____
Professor James W. Mar
Thesis Supervisor

Accepted by _____
Professor Harold Y. Wachman
Chairman, Departmental Graduate Committee

Abstract

Damage Resistance and Damage Tolerance of Thin Composite Facesheet Honeycomb Panels

by

Simon Charles Lie

Submitted to the Department of Aeronautics and Astronautics on March 2, 1989 in partial fulfilment of the requirements for the Degree of Master of Science

An experimental and analytical investigation was made into the behavior of sandwich panels with two- and three-ply woven fabric graphite/epoxy facesheets subjected to low-velocity impacts. The facesheets were made from AW193PW/3501-6, a plain wave fabric made from Hercules AS4 fibers and 3501-6 epoxy. The core material was 3.0 pcf Nomex. Damage was observed to begin at an energy level of 1.0 ft lbs. Damage consisted of core crushing, matrix cracking, fiber breakage, and delaminations. Panels were also subjected to static indentation which produced damage indistinguishable from impact damage. Damaged panels were then tested in compression to failure. Panels with (±45) facesheets were found to be notch-insensitive and failed by local facesheet buckling (also called "facesheet wrinkling") at a constant value of net-section stress. Panels with (0,90) facesheets were notch-sensitive and failed in the same manner at a lower value of net-section stress. Analytical models of the impact event and and compression test are able to predict the amount of damage caused and the residual strength of the damaged panel. These models comprise a fast, preliminary analysis tool for use in design of sandwich panels.

Thesis Supervisor: James W. Mar
Title: J.C. Hunsaker Professor of Aerospace Education

Acknowledgements

This document represents the work of many individuals. I would like to acknowledge some of many people who made this project possible. Thanks to Darrin Tebon and Sen-Hao Lai for the design and construction of the heavy parts of FRED. Thanks to Al Supple, who taught me many useful things, such as the ins and outs of three-phase wiring. Thanks to Ping Lee, who rarely fails us though we expect her to know everything. Thanks to all the graduate students, especially Kevin Saeger, Pierre Minguet, and Wilson Tsang for providing insight, encouragement, and a little sanity. Thanks to Professors Dugundji, Graves, and Lagace for their thoughts and guidance. Special thanks especially to Professor James Mar for his supervision, encouragement, and the freedom with which he allowed me to tackle this project.

And lastly, a very special thanks to three individuals whose assistance has been indispensable. They are responsible for the majority of the work as well as the moral of the team: Krista Beed, chief scriber; Dave Fleming, chief aligner and illustrator; and Rich Pemberton (the one with the short hair), chief data taker and doughnut maker.

This thesis is dedicated to Edie Erlanson and those from Camp Cebuano, whose support and encouragement has made this possible.

Foreword

This work was conducted at the Technology Laboratory for Advanced Composites (TELAC) in the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology. The work was sponsored by Boeing Helicopters under purchase order number TT 964237. The project monitor was Steve Llorente.

Table of Contents

List of Figures

List of Tables

List of Symbols

a	panel length
A, B	constants
A_{ij}	components of the laminate extensional stiffness matrix
$[\mathbf{A}]$	laminate extensional stiffness matrix
b	panel width
B_i	buckling load for mode shape i (residual strength model)
d	damage width
d_1, d_2	change over points in the corrected Mar-Lin relation (residual strength model)
D	plate bending stiffness
E	modulus of elasticity
E_{ij}	components of the material stiffness matrix
E_L	longitudinal modulus
E_T	transverse modulus
$[\mathbf{E}]$	material stiffness matrix
EI	bending stiffness of the beam
F	contact force acting on the panel
F_{max}	maximum contact force
F_{TU}	tensile strength
G	local minimum of strain energy at fixed value of λ
G_{LT}	shear modulus
h	core thickness
H_c	composite fracture toughness
i, j	dummy variables

I	area moment of inertia
k	Hertzian spring constant
$[\mathbf{K}]$	panel stiffness matrix
m	number of modes in the x-direction (global model)
m_0	impactor mass
$[\mathbf{M}]$	panel mass matrix
n	number of modes in the y-direction (global model)
$\tilde{n}$	number of modes (residual strength model)
N_1, N_2	linear shape functions
p, q	dummy variables
P	applied load
P_{cr}	buckling load
q_{kij}	time-varying generalized coordinates (global model)
$\dot{q}_{kij}$	time-varying generalized velocities (global model)
$\ddot{q}_{kij}$	time-varying generalized accelerations (global model)
$\bar{q}_i$	generalized coordinates (local model)
$\tilde{q}_i$	generalized coordinates (residual strength model)
$\{\mathbf{q}\}$	vector of generalized coordinates (global model)
$\{\dot{\mathbf{q}}\}$	vector of generalized velocities (global model)
$\{\ddot{\mathbf{q}}\}$	vector of generalized accelerations (global model)
r, s	dummy variables
$\bar{r}$	distance from impact point
Q_i, Q_{kij}	normalized modal forces (global model)
$\{\mathbf{Q}_i\}$	vectors of modal forces (global model)
t_a	thickness of facesheet A
t_b	thickness of facesheet B
T	kinetic energy of the panel (global model)

u_a, v_a, w_a	Cartesian displacements of facesheet A
$\dot{u}_a, \dot{v}_a, \dot{w}_a$	Cartesian displacements of facesheet A
u_b, v_b, w_b	Cartesian velocity of facesheet B
$\dot{u}_b, \dot{v}_b, \dot{w}_b$	Cartesian velocity of facesheet B
u_c, v_c, w_c	Cartesian velocity of the core
$\dot{u}_c, \dot{v}_c, \dot{w}_c$	Cartesian velocity of the core
V	potential energy of the panel (global model)
v	velocity vector
w_0	displacement of the impact site on the panel
W_{ext}	external work done on the panel by the impactor
$\{\mathbf{W}\}$	column of values of mode shapes evaluated at the impact site
x, y, z	Cartesian coordinates
z_a	z coordinate of the center of facesheet A
z_{ac}	z coordinate of the core-facesheet A interface
z_b	z coordinate of the center of facesheet B
z_{bc}	z coordinate of the core-facesheet B interface
α	indentation
α_{max}	maximum indentation
β	exponent in force-indentation relation
Δt	time step interval
δ_{ij}	Kronecker delta
ε_i	strains
ε_{ult}	failure strain
ϕ_i	Euler beam function mode shapes satisfying the x-direction boundary conditions (global model)
$\bar{\phi}_i$	mode shape (local model)
Φ	beam function mode shape integrals (x-directions)

κ	modulus of foundation (local model)
ℓ	beam length
λ	shape parameter
ν_{LT}	Poisson's ratio
π	3.14159...
Π	total potential energy of the system (residual strength model)
ρ	volume density
σ_o	buckling stress - undamaged specimen
σ_{cr}	buckling stress - damaged specimen
ω	areal density
ξ, η, ζ	non-dimensional Cartesian coordinates
ψ_j	Euler beam function mode shapes satisfying the y-direction boundary conditions (global model)
Ψ	beam function mode shape integrals (y-directions)

1. Introduction

In recent years, advanced composite structures have become the subject of investigation for use in aerospace structures. In particular, sandwich structures have shown promise for use in rotary-wing craft.[1] The relatively low static strength requirements coupled with stringent vibration (stiffness) requirements tend to favor light-weight honeycomb sandwich construction. In addition to strength and stiffness requirements, the structure must be able to withstand accidental impacts and foreign-object damage. Currently, there is no consistent methodology for the impact analysis of honeycomb sandwich structures. This project is an attempt to develop a consistent methodology to predict the damage resistance[2] and damage tolerance[3] of thin gauge composite honeycomb structures. Specifically, the goal is to analytically predict the residual compressive strength of an impacted sandwich panel.

The project consisted of two portions: an experimental investigation and analytic modeling. As both portions were conducted simultaneously, insight from the experiments helped

---

[1]The first all-composite helicopter, the Boeing Model 360, flew in June 1987.

[2]Damage resistance refers to the amount and type of damage caused by a specific impact event.

[3]Damage tolerance refers to the residual strength of a damaged structure.

shape the analysis and preliminary analytical results helped determine the experimental parameters.

The information in this report is organized as follows: Chapter 2 contains the theoretical analysis; Chapter 3 describes the computer implementation of the analysis; Chapter 4 describes the procedures used to manufacture and test the sandwich structures; Chapter 5 contains the results of both the analytic and experimental portions of the investigation; and Chapter 6 contains the conclusions and summary. FORTRAN source code listings and experimental and analytic data is contained in the three appendices.

2. Theoretical Analysis

2.1 Background

Many investigators have developed models of impact events. Cairns[4] at MIT and Graves and Koontz[5] at Boeing Advanced Systems have each developed models. Cairns developed a time-marching solution to the general problem of an impact to a laminated plate. Graves and Koontz developed a closed-form solution for the problem of an impact of an elastic body on a simply-supported or clamped orthotropic plate. Like Cairns' analysis, the present analysis is a time-marching method, but it is restricted to orthotropic sandwich structures with "thin" facesheets. These assumptions allow simplifications which reduce the computational effort required for a solution.

Like Cairns', the present analysis treats the issues of damage tolerance and damage resistance separately and independently. Each step in the analysis is a self-contained module which requires certain well-defined inputs and produces specific outputs. With this type of architecture,

---

[4]Douglas S. Cairns, Impact and Post-Impact Response of Graphite/Epoxy and Kevlar/Epoxy Structures, Ph.D. Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, August 1987.

[5]Michael J. Graves and Jan Koontz, Initiation and Extent of Impact Damage in Graphite/Epoxy and Graphite/PEEK Composites, AIAA 88-2327, Proceedings of the 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA, 1988, page 967.

one step in the analysis may be easily changed or upgraded without modification to the remaining processes.

Before describing the analysis, a note about nomenclature is required. The term "panel" shall be used to refer to a structure consisting of two facesheets bonded to a Nomex core. The terms "plate" and "beam" are used in the classical, abstract sense to refer to concepts used in the development of the model of the panel.

2.2 Analytical Approach

The present analysis addresses the problem of modeling a low-velocity impact of an object and a thin facesheet composite honeycomb sandwich panel. The analysis begins with an overall model of the impact event. The actual event is an interaction of two deformable bodies. Idealized models of the impact event will be formulated in sufficient detail to model the important physical aspects of the impact event. Figure 2.1, Analysis Overview, shows the flow of information through the different modules of the analysis.

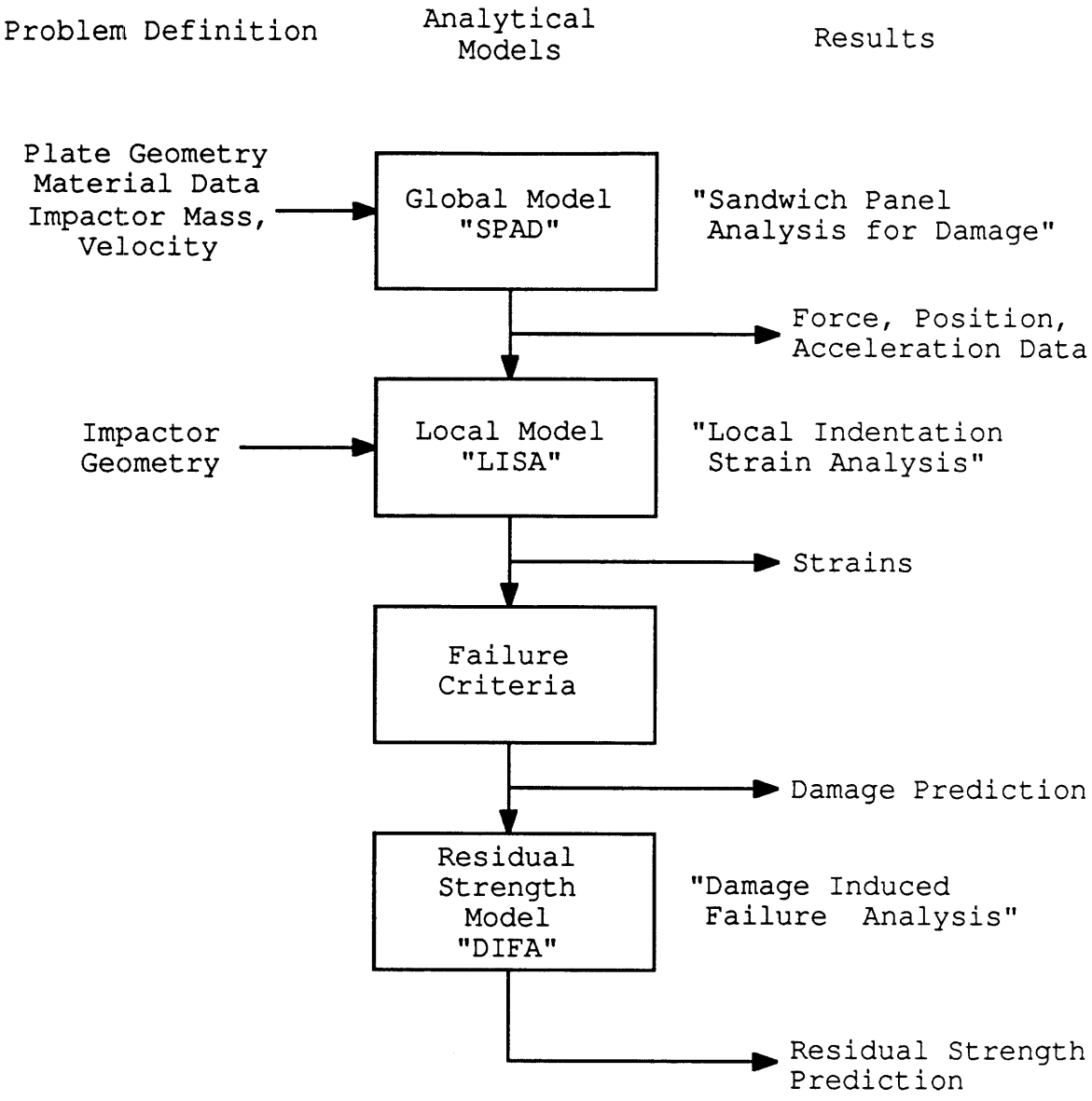


Figure 2.1 Analysis Overview

The impact event will be separated into two distinct models for analysis. The first model is an overall examination of the dynamics of the panel and impactor called the "global model." The specific global model developed here is known as "SPAD" - Sandwich Panel Analysis for Damage.

The results of the global model are used as input to the second model. This second model is a detailed examination of the local strains near the impact point and is thus called the "local model." The specific local model developed here is known as "LISA" - Local Indentation Strain Analysis. Together, the global and local models represent two separate physical phenomena: overall plate deflections and local deformations. Overall plate deflections are considered only in the global model. Local deformations are modeled by a Hertzian spring in the global model. In the local model, an energy method is used to analyze the local deformations. The following diagram is a graphical representation of the relationship between the two models and the actual impact event.

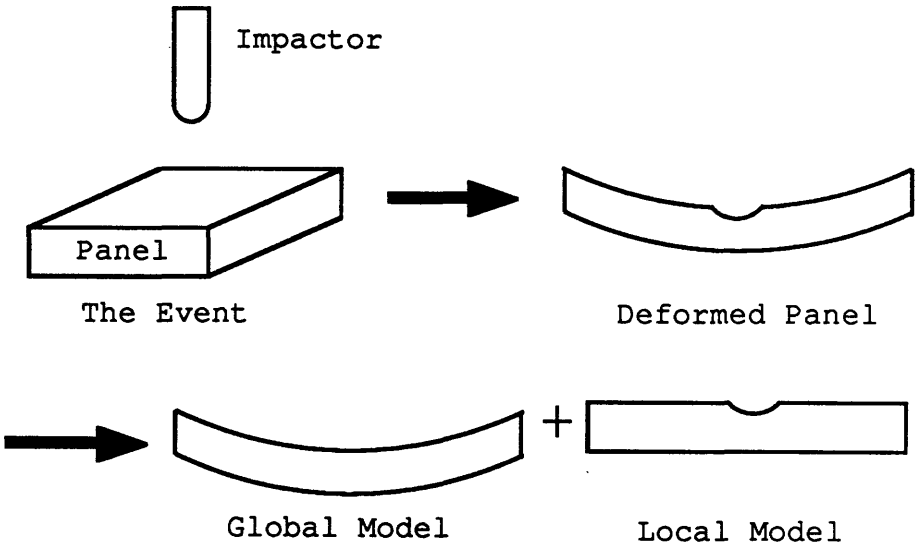


Figure 2.2 Modeling Method

2.3 Global Model

Let us now examine the global model. The geometry of the panel and impactor for the global model is defined as follows:

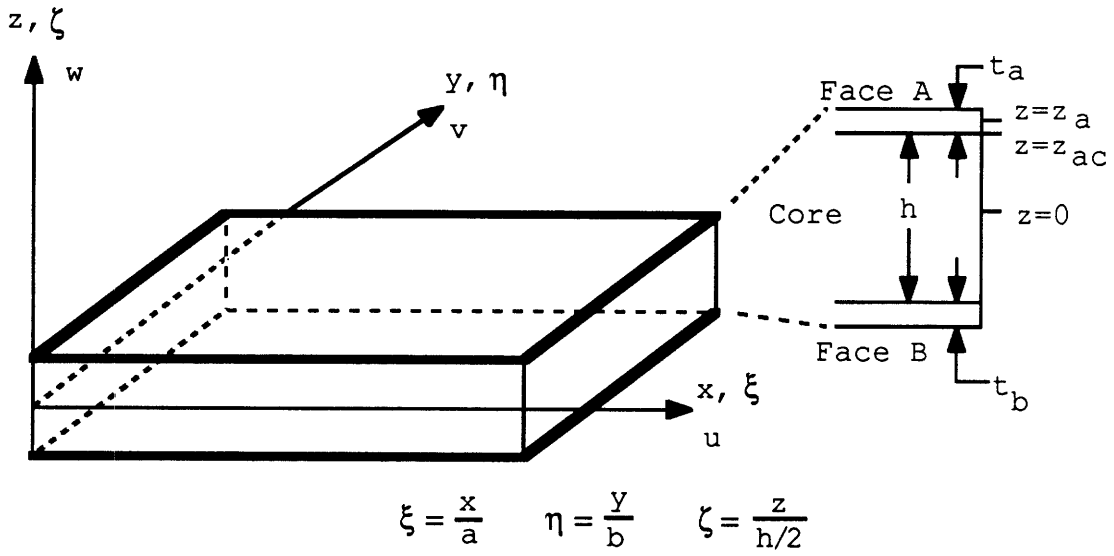


Figure 2.3 Global Model Panel Geometry

Any engineering model begins with assumptions and idealizations. The following assumptions are used for the formulation of the global model:

1. The sandwich panel is rectangular in shape and is composed of two uniform orthotropic facesheets perfectly bonded to a uniform thickness core.

2. The facesheets are thin relative to the core so that under bending, the stresses are assumed constant through the thickness of an individual facesheet (plane stress).

3. The core is "soft" in that its stiffness in the x-y plane is negligibly small compared to that of the facesheets.

4. The panel is ideally supported by any combination of free, simply-supported, or clamped conditions which prevent rigid body translation and rotation. Neither adjoining nor opposite sides are required to have the same boundary conditions.

5. The impactor is assumed to be rigid compared to the panel.

6. The strains in the panel will be small, that is to say, describable by linear strain-displacements relations.

7. Both the facesheets and the core are assumed to obey linear stress-strain relations.

8. The impact is assumed to follow the Hertzian Contact Law[6] for elastic bodies.

The problem is to find the displacements, and thus the strains, caused by an impact to the panel.

---

[6]S.P. Timoshenko and J.N. Goodier, Theory of Elastic Stability, Third Edition, McGraw-Hill, Inc., Singapore, 1982, page 409.

The displacements of the panel are modeled by 6 independent quantities: u_a, v_a, w_a, u_b, v_b, and w_b[7]. These represent the displacements of the two facesheets, A and B respectively. The core displacements are interpolated linearly from the facesheet displacements. Note that in this formulation, phenomena such as facesheet wrinkling and core crushing can be represented because the facesheets can deform independently.

In order to find a solution to this problem, the problem must be transformed from a continuous one to a discrete one. In this formulation, the displacements will be represented as a finite series of modes which are separable in time and space. These modes will be products of generalized beam functions[8,9] in the x and y directions which satisfy the specified boundary conditions. The beam functions shall be designated by $\phi_i(\xi)$ and $\psi_j(\eta)$. For brevity, the functional dependence will not be shown; q is a function of time only, ϕ is a function of ξ only, and ψ is a function of η only. Let m and n denote the number of modes in the x and y directions

---

[7]Here and elsewhere, the subscripts a, b, and c refer to facesheet A, facesheet B, and the core, respectively.

[8]John Dugundji, Simple Expressions for Higher Vibration Modes of Uniform Euler Beams, Technical Note, AIAA Journal, Volume 26, No. 8, August 1988, page 1013.

[9]Note: These beam functions do not exactly satisfy the boundary conditions. The worst error occurs in the lowest mode and is approximately 0.5% of the maximum amplitude.

respectively. There need not be the same number of modes in each direction. Also, because the facesheets are orthotropic, there is no bending-twisting coupling. A centerline impact will excite only symmetric modes and thus the anti-symmetric modes need not be included in the analysis.

The displacements are written:

$$w_a = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{1ij}\phi_i\psi_j \tag{1a}$$

$$u_a = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{2ij} z_a \phi'_i\psi_j \tag{1b}$$

$$v_a = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{3ij} z_a \phi_i\psi'_j \tag{1c}$$

$$w_b = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{4ij}\phi_i\psi_j \tag{1d}$$

$$u_b = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{5ij} z_b \phi'_i\psi_j \tag{1e}$$

$$v_b = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{6ij} z_b \phi_i\psi'_j \tag{1f}$$

where: ϕ_i and ψ_j are generalized beam functions which satisfy the prescribed boundary conditions,

q_{kij} are generalized displacements representing modal amplitudes,

and the prime notation denotes spatial partial differentiation.

In the modal amplitudes q_{kij}; k varies from 1-6 referring to w_a, u_a, v_a, w_b, u_b, or v_b respectively; i varies from 1 to m referring to the x direction mode number; and j varies from 1 to n referring to the y direction mode number.

Note that the displacements u and v, i.e., the in-plane mode shapes, are composed of w, i.e., the out-of-plane mode shapes. These in-plane modes where chosen so that the model could correctly represent pure bending of thin plates with no shear deformation. Under Kirchoff assumptions of thin plates with no mid-plane extension, the out-of-plane displacements imply certain in-plane displacements. These displacements are given by:

$$u = -z\frac{\partial w}{\partial x} \tag{2a}$$

$$v = -z\frac{\partial w}{\partial y} \tag{2b}$$

The right-hand-sides of the these equations were chosen to serve as the in-plane displacement modes. The negative sign was omitted from the mode shapes and becomes incorporated into the modal amplitude. If the six modal amplitudes, q_{kij}, are equal[10] for all modes (i.e. $q_{sij}=q_{tij}$ for s,t in {1..6}) then the displacement field represents pure

[10]Due to the inclusion of the negative sign in the amplitudes of the in-plane modes, the actual condition is $q_{1ij} = -q_{2ij} = -q_{3ij}$.

bending which satisfies the Kirchoff assumptions of no mid-plane extension and no shear deformation.

The core displacements are interpolated linearly from the facesheet displacements as follows:

$$w_c = N_1 w_a + N_2 w_b \tag{3a}$$

$$u_c = N_1 u_a + N_2 u_b \tag{3b}$$

$$v_c = N_1 v_a + N_2 v_b \tag{3c}$$

where the shape functions N_1 and N_2 are defined as:

$$N_1 = \frac{1}{2}(1+\zeta) \qquad N_2 = \frac{1}{2}(1-\zeta) \tag{4a,b}$$

where: $\zeta = \frac{z}{h/2}$

Substituting the expressions for the facesheet displacements (equations 1a-1f) into the above equations gives the core displacements in terms of the modal amplitudes:

$$w_c = \sum_{i=1}^{m}\sum_{j=1}^{n}(N_1 q_{1ij} + N_2 q_{4ij})\,\phi_i\,\psi_j \tag{5a}$$

$$u_c = \sum_{i=1}^{m}\sum_{j=1}^{n}(N_1 q_{2ij} z_{ac} + N_2 q_{5ij} z_{bc})\,\phi_i'\,\psi_j \tag{5b}$$

$$v_c = \sum_{i=1}^{m}\sum_{j=1}^{n}(N_1 q_{3ij} z_{ac} + N_2 q_{6ij} z_{bc})\,\phi_i\,\psi_j' \tag{5c}$$

The modal amplitudes q_{kij} are the generalized coordinates of the problem. With this formulation, as many modes as

desired may be used, but generally 3 or 4 in each direction is sufficient. The number of generalized coordinates is the number of unknowns and is equal to (6 x m x n).

Because the facesheets are assumed to be thin relative to the panel, each facesheet is assumed to be in a state of plane stress. Thus, all quantities are assumed constant through the thickness of each facesheet. Also, the core material (Nomex) is very soft in the x and y directions relative to the facesheets and thus its stiffness in these directions is assumed to be zero. With these assumptions, the interesting strains in the panel are ε_1, ε_2, and ε_6 in the facesheets and ε_3, ε_4, and ε_5 in the core. Because the overall strains in the panel are expected to be small, linear strain-displacement relations are used to write:

In the facesheets:

$$\varepsilon_1 = \frac{\partial u}{\partial x} \tag{6a}$$

$$\varepsilon_2 = \frac{\partial v}{\partial y} \tag{6b}$$

$$\varepsilon_6 = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{6c}$$

In the core:

$$\varepsilon_3 = \frac{\partial w}{\partial z} \tag{7a}$$

$$\varepsilon_4 = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \tag{7b}$$

$$\varepsilon_5 = \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \tag{7c}$$

Note: Engineering rather than tensor shear strain is used.

Substituting the expressions for the displacements (equations 1 and 5) into the expressions for the the strains (equations 6 and 7) gives expressions for the strains in terms of the modal amplitudes q_{kij}:

In face A:

$$\varepsilon_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{2ij} z_a \phi''_i \psi_j \tag{8a}$$

$$\varepsilon_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{3ij} z_a \phi_i \psi''_j \tag{8b}$$

$$\varepsilon_6 = \sum_{i=1}^{m}\sum_{j=1}^{n} (q_{2ij} + q_{3ij}) z_a \phi'_i \psi'_j \tag{8c}$$

In face B:

$$\varepsilon_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{5ij} z_b \phi''_i \psi_j \tag{9a}$$

$$\varepsilon_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{6ij} z_b \phi_i \psi''_j \tag{9b}$$

$$\varepsilon_6 = \sum_{i=1}^{m}\sum_{j=1}^{n} (q_{5ij} + q_{6ij}) z_b \phi'_i \psi'_j \tag{9c}$$

In the core:

$$\varepsilon_3 = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{1}{h} (q_{1ij} - q_{4ij}) \phi_i \psi_j \tag{10a}$$

$$\varepsilon_4 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left[(N_1 q_{1ij} - N_2 q_{4ij}) + \frac{1}{h}(q_{3ij} z_{ac} - q_{6ij} z_{bc})\right]\phi_i \psi'_j \qquad (10b)$$

$$\varepsilon_5 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left[(N_1 q_{1ij} - N_2 q_{4ij}) + \frac{1}{h}(q_{2\,ij} z_{ac} - q_{5\,ij} z_{bc})\right]\phi'_i \psi_j \qquad (10c)$$

The equations of motion of the panel are found using Lagrange's Equations[11]:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = F\,Q_i \qquad (11)$$

where: T is the kinetic energy of the panel,

V is the potential energy of the panel,

F is a scalar representing the total force applied to the panel,

and Q_i are normalized modal forces associated with the generalized displacements q_i.

Our goal, then, is to express the kinetic and potential energy of the panel in terms of the generalized displacements. With this in mind, let us begin.

Potential Energy

The potential energy of the panel is given by:

$$V = \frac{1}{2}\int_{\text{panel}}\int\int \{\varepsilon\}^T [\mathbf{E}]\{\varepsilon\}\, dx\, dy\, dz \qquad (12)$$

---

[11]Raymond L. Bisplinghoff and Holt Ashley, Principles of Aeroelasticity, Dover Publications, Inc., New York, 1962, page 35.

Breaking this integral into separate regions covering the facesheets and core gives:

$$V = \frac{1}{2}\int\int\int_{\text{face A}} \{\varepsilon_a\}^T [\mathbf{E}_a]\{\varepsilon_a\}\, dx\, dy\, dz + \frac{1}{2}\int\int\int_{\text{face B}} \{\varepsilon_b\}^T [\mathbf{E}_b]\{\varepsilon_b\}\, dx\, dy\, dz + \frac{1}{2}\int\int\int_{\text{core}} \{\varepsilon_c\}^T [\mathbf{E}_c]\{\varepsilon_c\}\, dx\, dy\, dz \quad (13)$$

where:

$$\{\varepsilon_a\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_6 \end{Bmatrix}_a \qquad \{\varepsilon_b\} = \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_6 \end{Bmatrix}_b \qquad \{\varepsilon_c\} = \begin{Bmatrix} \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{Bmatrix}_c \quad (14a,b,c)$$

and

$$[\mathbf{E}_a] = \begin{bmatrix} E_{11} & E_{12} & 0 \\ E_{12} & E_{22} & 0 \\ 0 & 0 & E_{66} \end{bmatrix}_a \quad (15a)$$

$$[\mathbf{E}_b] = \begin{bmatrix} E_{11} & E_{12} & 0 \\ E_{12} & E_{22} & 0 \\ 0 & 0 & E_{66} \end{bmatrix}_b \quad (15b)$$

$$[\mathbf{E}_c] = \begin{bmatrix} E_{33} & 0 & 0 \\ 0 & E_{44} & 0 \\ 0 & 0 & E_{55} \end{bmatrix}_c \quad (15c)$$

The assumption of plane stress within each facesheet allows us to use classical laminated plate theory[12] to write:

$$[\mathbf{A}] = \int [\mathbf{E}]\, dz = \sum_{i=1}^{n_{ply}} [\mathbf{E}]^{[-\theta_i]} t_{ply} \tag{16}$$

Thus, equation 13 becomes:

$$\begin{aligned} V = & \frac{1}{2} \int\int_{\text{face A}} \{\varepsilon_a\}^T [\mathbf{A}_a] \{\varepsilon_a\} dx\, dy \\ & + \frac{1}{2} \int\int_{\text{face B}} \{\varepsilon_b\}^T [\mathbf{A}_b] \{\varepsilon_b\} dx\, dy \\ & + \frac{1}{2} \int\int\int_{\text{core}} \{\varepsilon_c\}^T [\mathbf{E}_c] \{\varepsilon_c\} dx\, dy\, dz \end{aligned} \tag{17}$$

Substituting equations 14 and 15 into 17 yields:

$$\begin{aligned} V = & \frac{1}{2} \int\int_{\text{face A}} (A_{11}\varepsilon_1^2 + 2A_{12}\varepsilon_1\varepsilon_2 + A_{22}\varepsilon_2^2 + A_{66}\varepsilon_6^2)\, dx\, dy \\ & + \frac{1}{2} \int\int_{\text{face B}} (A_{11}\varepsilon_1^2 + 2A_{12}\varepsilon_1\varepsilon_2 + A_{22}\varepsilon_2^2 + A_{66}\varepsilon_6^2)\, dx\, dy \\ & + \frac{1}{2} \int\int\int_{\text{core}} (E_{33}\varepsilon_3^2 + E_{44}\varepsilon_4^2 + E_{55}\varepsilon_5^2)\, dx\, dy\, dz \end{aligned} \tag{18}$$

By substituting the expressions for the strains in terms of generalized displacements into this equation, we have the desired expression for potential energy in terms of generalized displacements. For brevity, we will show only

---

[12]Robert M. Jones, Mechanics of Composite Materials, Scripta Book Company, Washington, DC, 1975, page 154.

the first of the 15 individual terms of the resulting equation (the example term is due to A_{11a}):

$$V = \frac{1}{2}A_{11}\left[\sum_{i=1}^{m}\sum_{j=1}^{n} q_{2ij} z_a \iint_{\text{face A}} \phi_i'' \psi_j \, da\right]\left[\sum_{r=1}^{m}\sum_{s=1}^{n} q_{2rs} z_a \iint_{\text{face A}} \phi_r'' \psi_s \, da\right] + \ldots \tag{19}$$

Note that the only x or y dependency in the equation is contained in the ϕ and ψ terms. The integrals of these terms over the panel can be taken separately and are written as:

$$\Phi(i,j,p,q) = \int_{\xi=0}^{1}\left[\frac{\partial^p \phi_i}{\partial \xi^p}\right]\left[\frac{\partial^q \phi_j}{\partial \xi^q}\right] d\xi \tag{20a}$$

$$\Psi(i,j,p,q) = \int_{\eta=0}^{1}\left[\frac{\partial^p \psi_i}{\partial \eta^p}\right]\left[\frac{\partial^q \psi_j}{\partial \eta^q}\right] d\eta \tag{20b}$$

where: i and j are dummy variables signifying mode shape number,

and p and q are dummy variables indicating which derivative of the mode shape is to be used.

Substituting for the integrals in equation 19, we have:

$$V = \frac{1}{2}abA_{11}z_a^2 \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{r=1}^{m}\sum_{s=1}^{n} q_{2ij} q_{2rs} \Phi(i,r,2,2)\, \Psi(j,s,0,0) \quad + \quad \ldots \tag{21}$$

This procedure produces all 15 terms for the potential energy V. Note that all terms are quadratic in q_{kij}. When the partial differentiation called for by Lagrange's Equation (equation 10) is performed, these terms become linear in q_{kij}. Rewriting these equations in matrix notation results in:

$$\frac{\partial V}{\partial q_{imn}} = [\mathbf{K}] \begin{Bmatrix} \{q_{1mn}\} \\ \{q_{2mn}\} \\ \{q_{3mn}\} \\ \{q_{4mn}\} \\ \{q_{5mn}\} \\ \{q_{6mn}\} \end{Bmatrix} = [\mathbf{K}]\{\mathbf{q}\} \tag{22}$$

where [**K**] is a 6mn by 6mn matrix.

Kinetic Energy

The kinetic energy due to the motion of the panel is given by:

$$T = \frac{1}{2} \int_{\text{panel}} \int \int \rho(\vec{v} \bullet \vec{v}) dx\, dy\, dz \tag{23}$$

where: $\vec{v}$ is the vector velocity,

and ρ is the material density.

Again, we break this integral into separate regions:

$$\begin{aligned} T = & \frac{1}{2} \int_{\text{face A}} \int \int (\rho_a + \rho_f)(\vec{v} \bullet \vec{v})\, dx\, dy\, dz \\ & + \frac{1}{2} \int_{\text{face B}} \int \int (\rho_b + \rho_f)(\vec{v} \bullet \vec{v})\, dx\, dy\, dz \\ & + \frac{1}{2} \int_{\text{core}} \int \int \rho_c (\vec{v} \bullet \vec{v})\, dx\, dy\, dz \end{aligned} \tag{24}$$

where ρ_f represents the density of the film adhesive bonding the facesheet to the core.

The dot product $\vec{v} \bullet \vec{v}$ may be written as:

$$\vec{v} \bullet \vec{v} = (\dot{u}^2 + \dot{v}^2 + \dot{w}^2) \tag{25}$$

Also, the assumption of plane stress within the facesheets allows us to perform the z direction integration separately as:

$$\int \rho \, dz = \omega \tag{26}$$

where ω is the areal density.

Substituting equations 25 and 26 into equation 24 yields:

$$\begin{aligned} T = & \frac{1}{2} \iint_{\text{face A}} (\omega_a + \omega_f)(\dot{u}_a^2 + \dot{v}_a^2 + \dot{w}_a^2)\, dx\, dy \\ & + \frac{1}{2} \iint_{\text{face B}} (\omega_a + \omega_f)(\dot{u}_b^2 + \dot{v}_b^2 + \dot{w}_b^2)\, dx\, dy \\ & + \frac{1}{2} \iiint_{\text{core}} \rho_c (\dot{u}_c^2 + \dot{v}_c^2 + \dot{w}_c^2)\, dx\, dy\, dz \end{aligned} \tag{27}$$

In the expressions for u, v, and w (equations 3a-f), the only time-dependency is the q_{kij} terms. By differentiating the expressions for the displacements with respect to time, we obtain the expressions for the velocities:

$$\dot{w}_a = \sum_{i=1}^{m} \sum_{j=1}^{n} \dot{q}_{1ij} \phi_i \psi_j \tag{28a}$$

$$\dot{u}_a = \sum_{i=1}^{m} \sum_{j=1}^{n} \dot{q}_{2ij} z_b \phi'_i \psi_j \tag{28b}$$

$$\dot{v}_a = \sum_{i=1}^{m} \sum_{j=1}^{n} \dot{q}_{3ij} z_a \phi_i \psi'_j \tag{28c}$$

$$\dot{w}_b = \sum_{i=1}^{m}\sum_{j=1}^{n}\dot{q}_{4ij}\phi_i\psi_j \tag{28d}$$

$$\dot{u}_b = \sum_{i=1}^{m}\sum_{j=1}^{n}\dot{q}_{5ij}z_b\phi'_i\psi_j \tag{28e}$$

$$\dot{v}_b = \sum_{i=1}^{m}\sum_{j=1}^{n}\dot{q}_{6ij}z_b\phi_i\psi'_j \tag{28f}$$

These expressions may be substituted directly into equation 27. As with the potential energy, the integrals of ϕ and ψ are replace by Φ and Ψ (equations 20a-b). Again, for brevity, only the first term is shown:

$$T = \frac{1}{2}ab(\rho_a + \rho_f)\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{r=1}^{m}\sum_{s=1}^{n}(\dot{q}_{1ij}\dot{q}_{1rs}\Phi(i,r,0,0)\,\Psi(n,s,0,0)) + \ldots \tag{29}$$

These terms are all quadratic in $\dot{q}_{kij}$ (the generalized velocities). The differentiation with respect to the $\dot{q}_{kij}$ called for by Lagrange's Equations produces terms linear in $\dot{q}_{kij}$. The time differentiation then produces terms linear in $\ddot{q}_{kij}$. Using matrix notation, the kinetic energy may be written:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_{kij}}\right) = [\mathbf{M}]\begin{Bmatrix}\{\ddot{q}_{1ij}\}\\ \{\ddot{q}_{2ij}\}\\ \{\ddot{q}_{3ij}\}\\ \{\ddot{q}_{4ij}\}\\ \{\ddot{q}_{5ij}\}\\ \{\ddot{q}_{6ij}\}\end{Bmatrix} = [\mathbf{M}]\{\ddot{\mathbf{q}}\} \tag{30}$$

Generalized Forces

Because the impact force is assumed to act only in the z direction with no surface shears, the generalized forces Q_i are defined as:

$$F\{Q_i\} = F\begin{Bmatrix} \{Q_{1ij}\} \\ \{0\} \\ \{0\} \\ \{0\} \\ \{0\} \\ \{0\} \end{Bmatrix} \tag{31}$$

in which the only non-zero components are:

$$\{Q_{1ij}\} = \iint_{\text{face A}} p_z(x,y)\,\phi_i\,\psi_j\,dx\,dy \tag{32}$$

where: $p_z(x,y)$ represents the normalized force distribution produced by the impact event.

Note: p_z has units of in^{-2}.

As this integration is done numerically, any shape may be chosen for the function p_z. The only restriction is that the force distribution be normalized such that:

$$\iint_{\text{panel}} p_z(x,y)\,dx\,dy = 1 \tag{33}$$

For this study, a square load patch is used with total area equal to the area of the impactor. Within this load patch, the load is assumed constant.

The equations of motion of the panel are now complete.

Impactor

The equation of motion of the impactor is given by:

$$m_0 \ddot{q}_0 = -F \tag{34}$$

where: m_0 is the mass of the impactor,

q_0 is the position of the impactor,

and F is the force applied to the panel.

Additionally, we have the Hertzian contact law which couples the two sets of equations.

$$F = k\alpha^{3/2} \tag{35}$$

where: F is the contact force,

α is the approach (see Figure 2.4),

and k is the Hertzian spring constant.

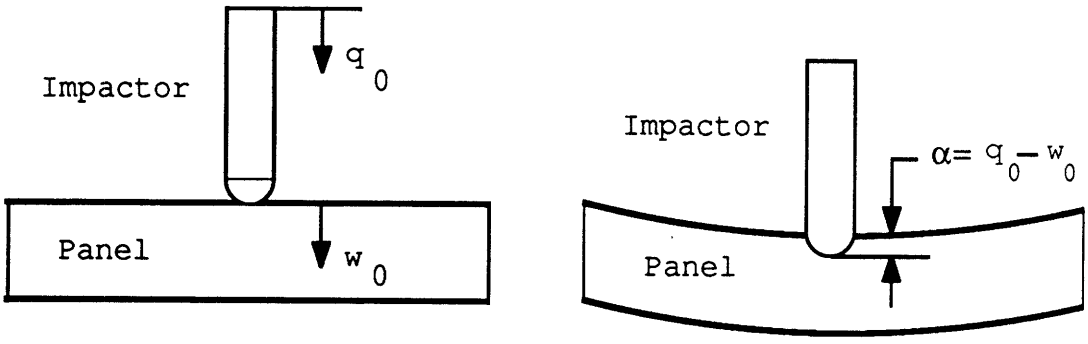


Figure 2.4 Impact Geometry

From Figure 2.4, it can be seen that the approach α may be written as q_0-w_0. Expressing w_0 in generalized coordinates gives:

$$w_0 = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{1ij}\phi_i(\tfrac{1}{2})\psi_j(\tfrac{1}{2}) = \{\mathbf{W}\}^T\{\mathbf{q}\} \tag{36}$$

so that the contact law (equation 35) may be written as:

$$F = k\alpha^{3/2} = k\left(q_0 - \sum_{i=1}^{m}\sum_{j=1}^{n} q_{1ij}\phi_i(\tfrac{1}{2})\,\psi_j(\tfrac{1}{2})\right)^{3/2} \tag{37}$$

or, in matrix notation:

$$F = k(q_0 - \{\mathbf{W}\}^T\{\mathbf{q}\})^{3/2} \tag{38}$$

We now have the following equations:

The equations of motion of the panel, in matrix form,

$$[\mathbf{M}]\{\ddot{\mathbf{q}}\} + [\mathbf{K}]\{\mathbf{q}\} = F\{\mathbf{Q}\} \tag{39}$$

the equation of motion of the impactor,

$$m_0\ddot{q}_0 = -F \tag{40}$$

and non-linear contact law which couples the motion of the panel and impactor (equation 38).

2.4 Newmark Integration

The equations of motion are ordinary differential equations in time and are solved by the Newmark constant

average acceleration scheme[13]. In this time-marching scheme, the acceleration is considered constant over the interval of one time step. For each acceleration, the value used is the average of the values at the beginning and end of the interval. The velocity is then calculated using this assumed value of acceleration. The position is calculated similarly from the velocity. The following diagram represents the scheme:

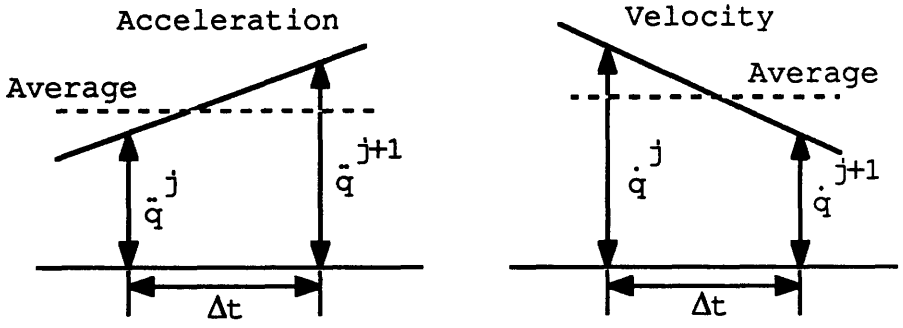


Figure 2.5 Newmark Integration Scheme

Mathematically:

$$\mathbf{q}^{j+1} = \mathbf{q}^{j} + \frac{\Delta t}{2}\left(\dot{\mathbf{q}}^{j} + \dot{\mathbf{q}}^{j+1}\right) \tag{41}$$

$$\dot{\mathbf{q}}^{j+1} = \dot{\mathbf{q}}^{j} + \frac{\Delta t}{2}\left(\ddot{\mathbf{q}}^{j} + \ddot{\mathbf{q}}^{j+1}\right) \tag{42}$$

where: q^{j} denotes the quantity q at time step j.

By eliminating $\dot{\mathbf{q}}^{j+1}$ from these equations and solving for $\ddot{\mathbf{q}}^{j+1}$ we have:

---

[13]K.J. Bathe, Numerical Methods in Finite Element Analysis, Prentice-Hall, Inc., New Jersey, 1982, page 511.

$$\ddot{\mathbf{q}}^{j+1} = \frac{4}{\Delta t^2}\left(\mathbf{q}^{j+1} - \mathbf{q}^{j} - \dot{\mathbf{q}}^{j}\Delta t\right) - \ddot{\mathbf{q}}^{j} \tag{43}$$

substituting (43) back into (42) gives:

$$\dot{\mathbf{q}}^{j+1} = \dot{\mathbf{q}}^{j} + \frac{2}{\Delta t}\left(\mathbf{q}^{j+1} - \mathbf{q}^{j} - \dot{\mathbf{q}}^{j}\Delta t\right) \tag{44}$$

Note that in this equation there are still quantities at the current time step as well as at the previous time step on the right-hand-side.

By substituting the new expression for the acceleration (equation 41) into equation 37, we can write the equation of motion of the panel at time step j+1.

$$[\mathbf{M}]\left\{\frac{4}{\Delta t^2}\left(\mathbf{q}^{j+1} - \mathbf{q}^{j} - \dot{\mathbf{q}}^{j}\Delta t\right) - \ddot{\mathbf{q}}^{j}\right\} + [\mathbf{K}]\{\mathbf{q}^{j+1}\} = F^{j+1}\{\mathbf{Q}\} \tag{45}$$

solving for $\mathbf{q}^{j+1}$ yields:

$$\mathbf{q}^{j+1} = \left[\frac{4}{\Delta t^2}[\mathbf{M}] + [\mathbf{K}]\right]^{-1}\left\{F^{j+1}\{\mathbf{Q}\} + [\mathbf{M}]\left\{\ddot{\mathbf{q}}^{j} + \frac{4}{\Delta t^2}\left(\mathbf{q}^{j} + \dot{\mathbf{q}}^{j}\Delta t\right)\right\}\right\} \tag{46}$$

We can also write the contact law (equation 38) at time j+1:

$$F^{j+1} = k\left(q_0^{j} + \Delta t\dot{q}_0^{j} + \frac{\Delta t^2}{4}\left(\ddot{q}_0^{j} - \frac{F^{j+1}}{m_0}\right) - \{\mathbf{W}\}^{T}\{\mathbf{q}^{j+1}\}\right)^{3/2} \tag{47}$$

We now have two equations (46 and 47) for two unknowns ($\mathbf{q}^{j+1}$ and F^{j+1}). Substituting equation 46 into equation 47 yields:

$$F^{j+1} = k\left[\begin{array}{l} q_0^j + \Delta t\dot{q}_0^j + \frac{\Delta t^2}{4}\left(\ddot{q}_0^j - \frac{F^{j+1}}{m_0}\right) - \\ \{W\}^T\left[\frac{4}{\Delta t^2}[M] + [K]\right]^{-1}\left\{F^{j+1}\{Q\} + [M]\left\{\ddot{q}^j + \frac{4}{\Delta t^2}\left(q^j + \dot{q}^j\Delta t\right)\right\}\right\} \end{array}\right]^{3/2} \tag{48}$$

This equation is now of the form:

$$F = k(AF + B)^{3/2} \tag{49}$$

with the following definitions for the scalars A and B:

$$A = -\frac{\Delta t^2}{4m_0} - \{W\}^T\left[\frac{4}{\Delta t^2}[M] + [K]\right]^{-1}\{Q\} \tag{50}$$

$$B = \left(q_0^j + \Delta t\dot{q}_0^j + \frac{\Delta t^2}{2}\ddot{q}_0^j\right) - \{W\}^T\left[\frac{4}{\Delta t^2}[M] + [K]\right]^{-1}[M]\left\{\ddot{q}^j + \frac{4}{\Delta t^2}\left(q^j + \Delta t\dot{q}^j\right)\right\} \tag{51}$$

We now have all the required equations to implement the solution. The procedure at each time step is as follows:

1. Form the quantities A and B (equations 50 and 51).

2. Solve equation 49 for F^{j+1}. Newton-Raphson iteration is used for this procedure. If the quantity (AF+B) in equation 49 is less than zero, the panel and impactor are no longer in contact and the force is set to zero.

3. Calculate $\mathbf{q}^{j+1}$ (equation 46).

4. Calculate $\dot{\mathbf{q}}^{j+1}$ (equation 44).

5. Calculate $\ddot{\mathbf{q}}^{j+1}$ (equation 43).

6. Calculate q_0^{j+1} (equation 40).

7. Calculate $\dot{q}^{j+1}$ (equation 41).

8. Calculate $\ddot{q}^{j+1}$ (equation 42).

This completes the calculation cycle. The time marching scheme is continued until two conditions are met:

1) The impactor loses contact with the panel, and

2) The impactor has rebounded past its original position.

In this way, the complete global response of the panel can be found. From the analysis, any desired quantities can be extracted, such as force-time history, position-time history, maximum deflections, maximum impact force, etc.

2.5 Local Model

From the global model, the point of maximum indentation (which is also the point of maximum force) is singled out for further study in the local model. The maximum indentation and maximum force are denoted α_{max} and F_{max} respectively. The local indentation problem is a static problem in which the back surface of the panel (facesheet B) remains fixed and an indentation of α_{max} is imposed at the center of the top of the panel (facesheet A).

Castigliano's theory of least work states that the correct stress state is the one which minimizes elastic

strain energy.[14] We shall then seek the stress/strain state which minimizes the elastic strain energy of the panel.

As with the global model, the displacements of the panel are modeled by 6 independent unknowns (w_a, u_a, v_a, w_b, u_b, and v_b). However, because we assume that facesheet B remains fixed, three of the displacements are immediately constrained to be zero. The remaining displacements are modeled by a single assumed mode separable in time and space as follows:

$$w_a = \bar{q}_1(t)\,\bar{\phi}(\xi,\eta) \tag{52a}$$

$$u_a = \bar{q}_2(t)\,z_a \frac{\partial}{\partial x}\bar{\phi}(\xi,\eta) \tag{52b}$$

$$v_a = \bar{q}_3(t)\,z_a \frac{\partial}{\partial y}\bar{\phi}(\xi,\eta) \tag{52c}$$

where: $\bar{\phi}(\xi,\eta)$ is the mode shape

and $\bar{q}_i(t)$ are the generalized coordinates.

As with the global model, this choice of mode shapes allows the Kirchoff no-shear bending condition to be satisfied for thin plates. Note that in the local model, a single assumed mode shape is used to model the displacements.

---

[14]Robert M. Rivello, Theory and Analysis of Flight Structures, McGraw-Hill, New York, 1969, page 120.

Mode Shape

Timoshenko solved a related problem in which an isotropic plate supported by an elastic foundation is subjected to a out-of-plane line load. The plate is considered to be infinitely long and of unit width.

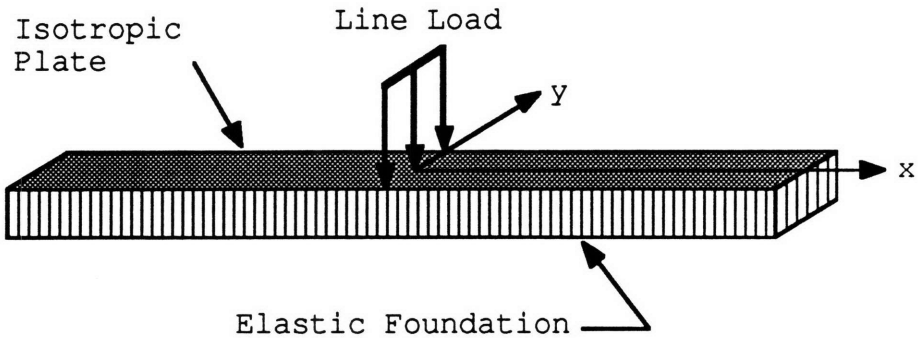


Figure 2.6 Plate on an Elastic Foundation

The plate is loaded at x=0 by a line load of P/unit width. Timoshenko shows that the deflections of the plate are given as[15]:

$$w(x) = -\frac{P\lambda}{2\sqrt{2}k} e^{\frac{-\lambda}{\sqrt{2}}x}\left[\cos\frac{\lambda}{\sqrt{2}}x + \sin\frac{\lambda}{\sqrt{2}}x\right] \qquad (53)$$

where: $\lambda^4 = \kappa/D$ is the shape parameter,

κ is the modulus of the foundation expressed as pressure exerted/unit deflection,

---

[15]S. P. Timoshenko and S. Woinowsky-Krieger, Theory of Plates and Shells, McGraw-Hill Book Company, Inc., New York, 1959, page 253.

and D is the plate bending stiffness.

Although this problem is not the same as ours, it can be considered similar. The top facesheet acts very much like a plate on an elastic foundation (the core). Therefore, we can expect the form of deflections in the local indentation problem to be similar to equation 53. Timoshenko's solution is for a line load, but our problem is a point load. If the facesheet and core were isotropic, then the indentation problem would be exactly axisymmetric. Although the facesheets and core are orthotropic, we shall assume an axisymmetric mode shape derived from equation 51. By rotating the above solution about the z-axis, the following axisymmetric mode shape is formed:

$$\bar{\phi}(\xi,\eta) = e^{-(\lambda\bar{r})}[\cos(\lambda\bar{r}) + \sin(\lambda\bar{r})] \tag{54}$$

where: λ is the shape parameter,

and $\bar{r} = \sqrt{\left(\xi - \frac{1}{2}\right)^2 + \left(\eta - \frac{1}{2}\right)^2}$

The shape parameter, λ, will be left as an unknown in the problem. The total number of unknowns is now 4: three generalized displacements ($\bar{q}_i$) and the shape parameter (λ).

The strains are formed from the displacements by using strain-displacement relations. In the facesheets, non-linear terms are included to account for the high strains near the impact region.

In facesheet A:

$$\varepsilon_1 = \frac{\partial u}{\partial x} + \frac{1}{2}\left[\left[\frac{\partial u}{\partial x}\right]^2 + \left[\frac{\partial v}{\partial x}\right]^2 + \left[\frac{\partial w}{\partial x}\right]^2\right] \tag{55a}$$

$$\varepsilon_2 = \frac{\partial v}{\partial y} + \frac{1}{2}\left[\left[\frac{\partial u}{\partial y}\right]^2 + \left[\frac{\partial v}{\partial y}\right]^2 + \left[\frac{\partial w}{\partial y}\right]^2\right] \tag{55b}$$

$$\varepsilon_6 = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial y} \tag{55c}$$

In the core:

$$\varepsilon_3 = \frac{\partial w}{\partial z} \tag{56a}$$

$$\varepsilon_4 = \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \tag{56b}$$

$$\varepsilon_5 = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \tag{56c}$$

The strain energy, U, is given by:

$$U = \frac{1}{2}\int_{\text{plate}}\int\int \{\varepsilon\}^T [E]\{\varepsilon\}\, dv \tag{57}$$

The formulation of the strain energy is similar to the formulation of the potential energy described earlier. The

only difference is that non-linear terms are included in the expressions for the strains. The resulting expression for strain energy is a function of 4 unknowns: $\bar{q}_1$, $\bar{q}_2$, $\bar{q}_3$, and λ.

The condition that the displacement of facesheet A match the indentation found using the global model imposes an additional constraint on the problem. Specifically,

$$w\left(\frac{1}{2},\frac{1}{2}\right) = \bar{q}_1 \bar{\phi}\left(\frac{1}{2},\frac{1}{2}\right) = \alpha_{max} \tag{58}$$

However, the value of the mode shape at the center of the plate (ξ=0.5, η=0.5) is independent of the shape parameter λ and is equal to 1. Thus, the indentation constraint can be written:

$$\bar{q}_1 = \alpha_{max} \tag{59}$$

This removes one degree-of-freedom from the model. The strain energy U is now a function of 3 variables. At this point, the values of $\bar{q}_1$, $\bar{q}_2$, and λ which minimize the strain energy must be found.

The calculation of the strain energy is much less involved if λ is held constant. Thus, the fewer times λ is varied, the faster the calculation will proceed. To take advantage of this characteristic, the function U is rewritten as follows:

$$U = G(\lambda) \tag{60}$$

where: $G(\lambda)$ is the minimum value of $U(\lambda, \bar{q}_1, \bar{q}_2)$ at fixed λ.

Note: a two-dimensional minimization occurs for each evaluation of $G(\lambda)$.

The two-dimensional minimization (over $\bar{q}_1$ and $\bar{q}_2$) is done using the Downhill Simplex Method[16]. The one-dimensional minimization over λ is done using Brent's Method[17] which combines bisection and parabola fitting. The result of this process is the value of the shape parameter and the generalized coordinates.

At this point, the entire local displacement field can be determined. As the global displacement field has already been determined, the total displacements can be found by summing the results from the local and global models. The total displacements are given by:

In Facesheet A:

$$w_a = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{1ij}\phi_i(\xi)\,\psi_j(\eta) + \bar{q}_1\bar{\phi}(\xi,\eta) \tag{61a}$$

$$u_a = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{2ij} z_a \phi'_i(\xi)\,\psi_j(\eta) + \bar{q}_2 z_a \frac{d}{dx}\bar{\phi}(\xi,\eta) \tag{61b}$$

---

[16]William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, Numerical Recipes - The Art of Scientific Computing, Cambridge University Press, Cambridge, 1986, page 289.

[17]Press et al, page 284.

$$v_a = \sum_{i=1}^{m} \sum_{j=1}^{n} q_{3ij} z_a \phi_i(\xi)\, \psi'_j(\eta) + \bar{q}_3 z_a \frac{d}{dy} \bar{\phi}(\xi,\eta) \tag{61c}$$

With the displacements known, the strains can be determined from the non-linear strain-displacement relations (equations 52 and 53). This portion of the analysis is now complete. From the properties of the panel and dynamics of the impactor, we have determined the strain field. This information is then be used for damage predictions.

2.6 Hertzian Spring Constant Determination

The local model can also be used to determine the value of the Hertzian spring constant, k, in the following expression:

$$F = k\alpha^{3/2} \tag{62}$$

This equation can be written more generally as:

$$F = k\alpha^{\beta} \tag{63}$$

where: β is a free parameter.

Note: For the Hertzian contact law, the parameter β is equal to 3/2.

Integrating both sides of equation 63 over the depth of the indentation yields an expression for the work done by the impactor:

$$W_{ext} = \int_0^{\alpha_{max}} F\, d\alpha = \int_0^{\alpha_{max}} k\alpha^{\beta}\, d\alpha = \frac{1}{(\beta+1)}\, k\alpha_{max}^{(\beta+1)} \tag{64}$$

where: W_{ext} is the work done on the panel by the impactor.

Note: In the local model, no global panel deformations are considered, so W_{ext} represents only the work done during the local deformation.

The external work is stored in the panel as strain energy and this quantity is already calculated by the local model. Several values of the strain energy (external work) are calculated - each for a different value of the indentation. The constants k and β are then found by a two-parameter least squares analysis. In general, β will not be equal to the Hertzian value of 3/2. Nevertheless, the value of 3/2 is used in the analysis. This is done for two reasons. First, the value 3/2 is theoretically justified; and second, the value of k is very sensitive to small variations in β.

WIth β fixed to a value of 3/2, equation 63 becomes a equivalent to the Hertzian contact law, equation 62. A second least squares analysis is done to find the corresponding value of k. This value of k is in general, different from the value found when β is a free parameter. This procedure can be done before the global model is run and thus the value of k determined can be used in the global model.

2.7 Failure Criteria

The purpose of finding the strain field caused by an impact event is to predict the amount and type of damage caused in the sandwich panel. Once the strains are determined, several failure criteria are available to predict the occurrence of damage. For this study, a simple point-strain criteria is used. This criteria states that failure (damage) occurs at any point at which the strain exceeds a specified critical value. Different strains cause different types of damage. In-plane extensional strains can be expected to cause fiber breakage. In-plane shear strains can be expected to cause matrix cracks. Out-of-plane extensional and shear strains are expected to cause delaminations. However, due to the assumption of plane stress, no out-of-plane information is generated and thus no predictions of delamination are possible.

2.8 Damage Predictions

For each strain in the impacted facesheet, a contour plot is made. On this plot, the contour corresponding to the failure strain serves as an outline of the predicted damage area. Different outlines are produces for different strains corresponding to different types of damage.

At this point, it should be noted that while the analysis method described here is used to predict damage,

both the global and local models assume that the panel remains undamaged throughout the impact event. A more complete analysis would update the properties of the panel as soon as damage is predicted in any area, but such an analysis is beyond the scope of this investigation.

2.9 Residual Strength Model

Once the damage area is known, the next step is to predict the residual strength. However, first we must analytically determine the undamaged strength of the panels.

The analytical failure loads are derived from the following model: the sandwich panel is modeled as a beam on an elastic foundation with the facesheet and film adhesive together acting as the beam and the core acting as the elastic foundation. For this specimen configuration, the film adhesive with embedded Nomex contributes a significant amount to the bending stiffness of the facesheet. Consequently, this effect must be included.

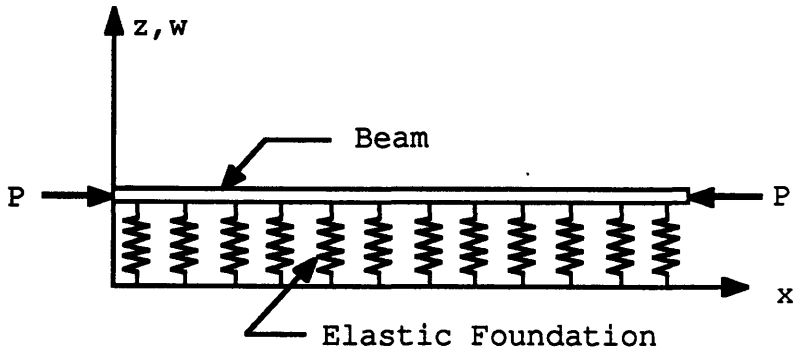


Figure 2.7 Compressive Strength Modeling

We shall assume that the beam is simply-supported at each end. This assumption will be justified later. Also, we shall assume that the mid-plane of the sandwich panel does not deform and that the failure mode is buckling of the beam. This phenomena is also known as "facesheet wrinkling."

The bending stiffness of the beam and modulus of the foundation are assumed to be constant. We shall use the principle of the stationary value of the total potential energy:

> Of all compatible states of deformation of an elastic body, the true deformations (those which satisfy equilibrium) are the ones for which the total potential energy has a stationary value for any virtual displacement.[18]

To implement this principle, we shall use the Rayleigh-Ritz method[19] of assumed modes. The displacements w are modeled as:

$$w = \sum_{i=1}^{\tilde{n}} \tilde{q}_i \sin \frac{n\pi x}{\ell} \tag{65}$$

where: $\tilde{n}$ is the total number of modes,

$\tilde{q}_i$ are the model amplitude (and thus generalized displacements),

ℓ is the length of the beam,

and x is the coordinate along the beam.

---

[18]Rivello, page 118.

[19]Rivello, page 123.

Note: These modes satisfy the geometric boundary conditions.

The total potential energy of the system is given by:

$$\Pi = U_{bending} + U_{spring} - W_{external} \tag{66}$$

$$\Pi = \frac{1}{2}\int_0^{\ell} EI\left(\frac{d^2w}{dx^2}\right)^2 dx + \frac{1}{2}\int_0^{\ell} \kappa w^2 dx - \frac{1}{2}\int_0^{\ell} P\left(\frac{dw}{dx}\right)^2 dx \tag{67}$$

where: κ is the modulus of foundation,

P is the applied end load,

and EI is the bending stiffness of the beam.

Substituting the assumed modes (equation 65) into the potential energy expression (equation 67) yields:

$$\begin{aligned}\Pi = {} & \frac{1}{2}EI\int_0^{\ell}\left[\sum_{i=1}^{\tilde{n}}\sum_{j=1}^{\tilde{n}} \tilde{q}_i\left(\frac{-i\pi}{\ell^2}\right)^2 \sin\frac{i\pi x}{\ell}\,\tilde{q}_j\left(\frac{-j\pi}{\ell^2}\right)^2 \sin\frac{j\pi x}{\ell}\right]dx \\ & + \frac{1}{2}\kappa\int_0^{\ell}\left[\sum_{i=1}^{\tilde{n}}\sum_{j=1}^{\tilde{n}} \tilde{q}_i \sin\frac{i\pi x}{\ell}\,\tilde{q}_j \sin\frac{j\pi x}{\ell}\right]dx \\ & - \frac{1}{2}P\int_0^{\ell}\left[\sum_{i=1}^{\tilde{n}}\sum_{j=1}^{\tilde{n}} \tilde{q}_i \frac{i\pi}{\ell}\sin\frac{i\pi x}{\ell}\,\tilde{q}_j \frac{j\pi}{\ell}\sin\frac{j\pi x}{\ell}\right]dx\end{aligned} \tag{68}$$

Due to orthogonality of modes, only those terms with i=j are non-zero:

$$\Pi = \frac{1}{2}EI\int_0^{\ell}\left[\sum_{i=1}^{\tilde{n}}\tilde{q}_i^2\left(\frac{-i\pi}{\ell^2}\right)^4 \sin^2\frac{i\pi x}{\ell}\right]dx + \frac{1}{2}\kappa\int_0^{\ell}\left[\sum_{i=1}^{\tilde{n}}\tilde{q}_i^2 \sin^2\frac{i\pi x}{\ell}\right]dx - \frac{1}{2}P\int_0^{\ell}\left[\sum_{i=1}^{\tilde{n}}\tilde{q}_i^2\left(\frac{i\pi}{\ell}\right)^2 \sin^2\frac{i\pi x}{\ell}\right]dx \tag{69}$$

Performing the integrals over the length of the beam yields:

$$\Pi = \sum_{i=1}^{\tilde{n}}\left\{EI\frac{i^4\pi^4}{4\ell^3} + \kappa\frac{\ell}{4} - P\frac{i^2\pi^2}{4\ell}\right\}\tilde{q}_i^2 \tag{70}$$

The virtual displacements are, in this case, the assumed modes themselves. For the total potential energy to remain stationary for these displacements requires that:

$$\frac{\partial\Pi}{\partial\tilde{q}_j} = \sum_{i=1}^{\tilde{n}}\left\{EI\frac{\pi^4}{4\ell^3}\left(i^4 + \frac{\kappa\ell^4}{\pi^4 EI}\right) - i^2\frac{P\pi^2}{4\ell}\right\}2\tilde{q}_i\delta_{ij} = 0 \tag{71}$$

where: δ_{ij} is the Kronecker Delta:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

The summation can now be eliminated. Rewriting equation 71 produces:

$$\frac{\partial\Pi}{\partial\tilde{q}_i} = \left\{EI\frac{\pi^2}{\ell^2}\left(i^2 + \frac{\kappa\ell^4}{i^2 EI\pi^4}\right) - P\right\}\tilde{q}_i = 0 \tag{72}$$

This equation is of the form:

$$(B_i - P)\tilde{q}_i = 0 \tag{73}$$

where: $$B_i = EI\frac{\pi^2}{\ell^2}\left(i^2 + \frac{\kappa\ell^4}{i^2 EI\pi^4}\right)$$

In matrix notation:

$$\begin{bmatrix} B_1 - P & 0 & \cdots & 0 \\ 0 & B_2 - P & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & B_{\tilde{n}} - P \end{bmatrix} \begin{Bmatrix} \tilde{q}_1 \\ \tilde{q}_2 \\ \vdots \\ \tilde{q}_{\tilde{n}} \end{Bmatrix} = \{0\} \tag{74}$$

This equation will only have non-trivial solutions if the determinant of the diagonal matrix is zero. This gives:

$$(B_1 - P)(B_2 - P)\ldots(B_{\tilde{n}} - P) = 0 \tag{75}$$

From equation 74, it can be seen that the mode which causes instability at the lowest load is the mode with the lowest associated value of B_i. This gives the buckling load:

$$P_{cr} = \min \frac{\pi^2 EI}{\ell^2}\left(i^2 + \frac{\kappa\ell^4}{i^2 EI\pi^4}\right) \tag{76}$$

where: P_{cr} is the buckling or facesheet wrinkling load.

We now have the failure load (or stress) for undamaged panels. For typical values of EI and κ, the mode with the lowest buckling load is in the range of i=12 to i=20. For these short wavelength modes, the effects of end boundary conditions (simply-supported, clamped, etc.) are negligible.

From equation 75, we know the failure load of undamaged panels. This load is converted to an equivalent stress and is referred to as σ_0. This stress is used as a basis to predict the residual strength of a damaged panel. Two separate methods are used to calculate the residual strength of a damaged panel: constant net-section stress (upper bound) and the Mar-Lin relation[20] (lower bound). If the panel is notch-insensitive, then the failure stress can be found by:

$$\sigma_{cr} = \sigma_0 \frac{(b-d)}{d} \tag{77}$$

where: σ_{cr} is the failure stress of the damaged panel,

σ_0 is the failure stress of an undamaged panel,

b is the panel width,

and d is the width of the damage area.

If the panel is notch-sensitive, then the Mar-Lin relation is used to predict the failure stress:

$$\sigma_{cr} = H_c(d)^{-0.28} \tag{78}$$

where: H_c is the experimentally determined composite "fracture toughness."

Note that as d approaches zero, equation 78 predicts that the failure stress will increase without bound. Also,

---

[20]K.Y. Lin, Fracture of Filamentary Composite Materials, Ph.D. Thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 1977.

as d approaches b, the predicted failure stress does not approach zero. Thus, corrections must be made near d/b=0 and near d/b=1. Consider a graph of equation with d/b as the abscissa and σ_{cr} as the ordinate. A straight line is constructed which passes through the point (d/b=0, $\sigma_{cr}=\sigma_0$) and is tangent to the curve (equation 78). Let d_1 denote the value of d at the point of tangency. Similarly, a second line is constructed which passes through the point (d/b=1, $\sigma_{cr}=0$) and is also tangent to the curve[21]. Let d_2 denote the value of d at the new point of tangency. Figure 2.8 depicts the Mar-Lin curve and tangent lines.

[21]Tangent line corrections are used by C. E. Federson in the study of cracked tension panels (metallic). C. E. Federson, Evaluation and Prediction of the Residual Strength of Center Cracked Tension Panels, ASTM Special Technical Publication 486, 1970, page 50.

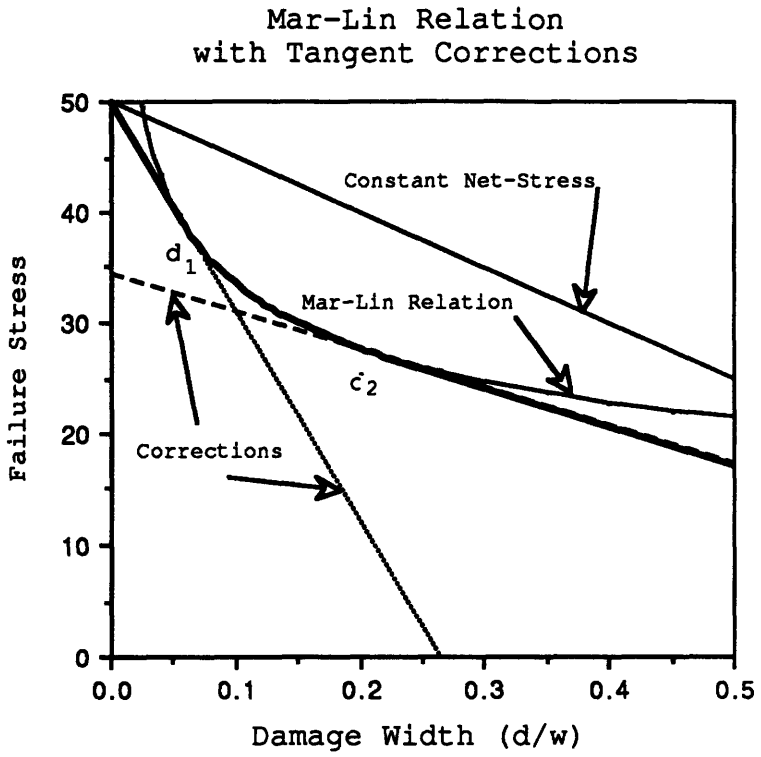


Figure 2.8 Mar-Lin Relation with Corrections

For values of d between zero and d_1, the upper tangent line is used to predict the failure stress. For values between d_1 and d_2, the Mar-Lin curve is used, and for values between d_2 and b, the lower tangent line is used. These sections of the various curves appear as the dark line in Figure 2.8.

For all panels, the two methods (net-section and Mar-Lin) are used predict the upper and lower bound of the residual strength. Panels with fibers perpendicular to the principal load axis are generally notch-sensitive and the value of residual strength lies closer to the lower bound. Panels with fibers at 45° to the principal load axis tend to

be notch-insensitive and the residual strength approaches the upper bound.

3. Computer Implementation

The analyses described in Chapter 2 have been implemented as FORTRAN programs on an IBM AT compatible computer. Microsoft's FORTRAN 4.1 Optimizing Compiler was used to compile the codes. Figure 3.1, Computer Model Overview, shows the different programs and associated data files.

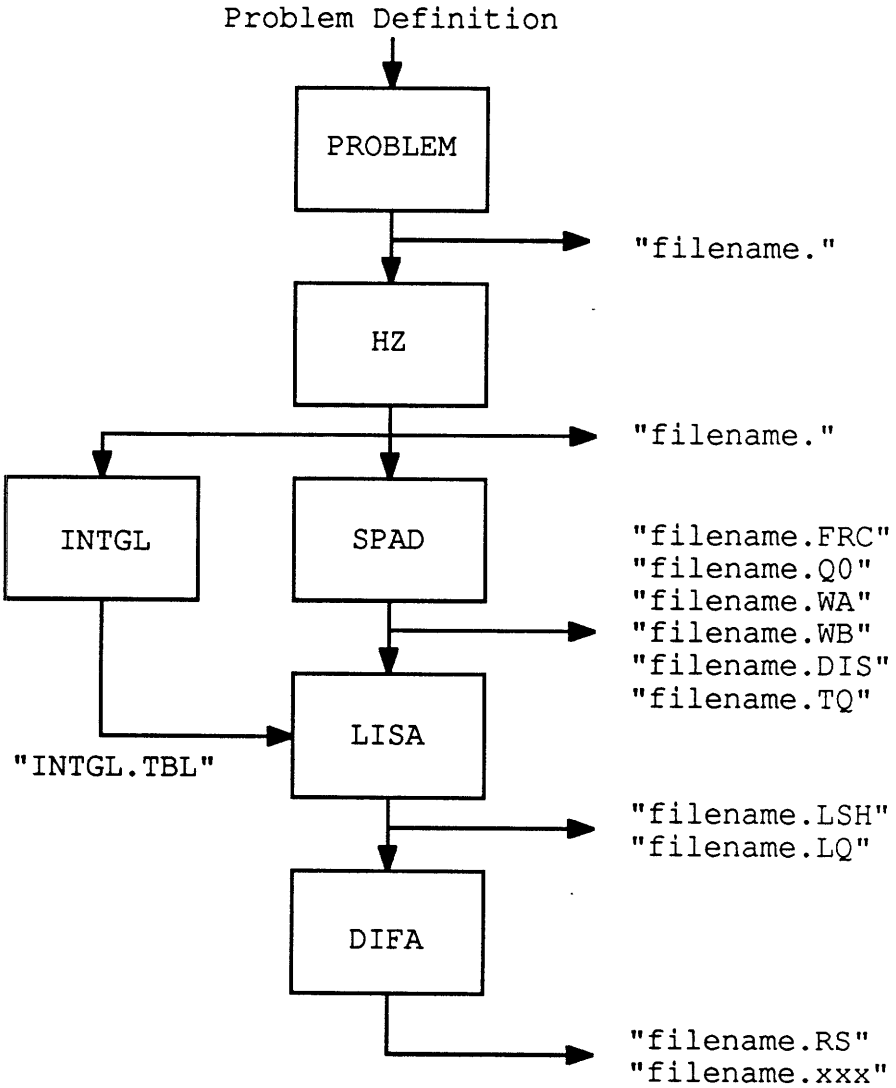


Figure 3.1 Computer Model Overview

The steps for use of the computer programs are as follows:

1. Type "PROBLEM". This program will ask questions about the panel and impact event to be analyzed as well as a filename. The filename may be up to 8 characters long. This filename shall be used by all other programs to refer to the same panel. The output from this program is a file with the given name and no extension. As an example, we shall use the name "filename".

   Input: Keyboard entry of panel and impact event information.

   Output: "filename." Problem description file.

2. Type "HZ filename". This executes the Hertzian spring constant determination routine derived from the local model. The results are stored in the problem description file.

   Input: "filename." Problem description file.

   Output: Updated "filename." Problem description file.

3. Type "SPAD filename". This executes the global dynamic analysis.

   Input: "filename." Problem description file.

   Output: "filename.FRC" Force-time history of the impact event.

   "filename.Q0" Displacement-time history of the impactor.

   "filename.WA" Displacement-time history of the center of the upper (impacted) facesheet.

   "filename.WB" Displacement-time history of the center of the lower facesheet.

   "filename.DIS" Displaced shape of the centerline of the upper facesheet at the instant of maximum load.

"filename.TQ" Data file of information required by program "LISA".

4. Type "INTGL filename". This executes the routines which generate the mode shape integrals needed by the local model. This step need not be repeated every time the analysis is run. It need only be run once.

   Input: "filename." Problem description file.

   Output: "INTGL.TBL" Table of integral values needed by the local model.

5. Type "LISA filename". This executes the local model.

   Input: "filename." Problem description file.

   "filename.TQ" Results from the global model.

   "INTGL.TBL" Table of integral values.

   Output: "filename.LQ" Results of the local model to be passed to "DIFA".

   "filename.LSH" Final displacement shape of the centerline of the impacted facesheet of the panel at the instant of maximum force including information from both the global and local models.

6. Type "DIFA filename". This executes the damage prediction and residual strength models. If desired, it will also produce contour plots of extensional strains in the top facesheet in any axes system in the plane.

   Input: "filename." Problem description file.

   "filename.TQ" Results of the global model.

   "filename.LQ" Results of the local model.

   Output: "filename.RS" Residual strength report.

   "filename.xxx" Contour plot files. These files are given extensions by the user.

Execution times for the various programs are as follows:

Table 3.1 Program Execution Times

	Program	Execution Time (Approximate)
Setup Programs:	HZ	3 minutes
	INTGL	8 minutes
Analysis Programs:	SPAD	45 seconds
	LISA	30 seconds
	DIFA	30 seconds

Note: The times listed above are for execution of the programs on a 16 MHz 80386 computer without numerical co-processor.

4. Experimental Procedures

4.1 Test Objectives

The objectives of the experimental tests were as follows:

1. To determine the material properties of the materials and the finished panels.
2. To characterize the types of damage caused by the low-velocity impacts.
3. To examine the relationship between impact energy and the amount of damage.
4. To examine the effect of this damage on the compressive strength of the panels.

Four test matrices have been created to accomplish these goals. The four test matrices and their specific objectives are:

Test Matrix S0: Material Properties

Test Matrix S1: Impact Damage - Two-Ply Facesheets

Test Matrix S2: Static Indentation - Two-Ply Facesheets

Test Matrix S3: Impact Damage - Three-Ply Facesheets

Test Matrix S0 established basic tension and compression properties of undamaged specimens. The purpose this test matrix is to verify the material properties of the panels and thus qualify the manufacturing procedures. As shown in Table 4.1, a total of 33 specimens are tested in Test Matrix S0. Parentheses and commas are used in the facesheet layup

notation to signify that this is a woven cloth material[22]. The angle specified is the orientation of the warp fibers. Thus, a +45° ply has warp fibers oriented at +45° and weft fibers oriented at -45°.

Table 4.1 Test Matrix S0

Facesheet Layup	Core Thickness	Load Type	Number of Specimens
(0_2)	none	tension	8
(90_2)	none	tension	8
(± 45)	none	tension	8
(0_2)	0.687 in	compression	3
(90_2)	0.687 in	compression	3
(± 45)	0.687 in	compression	3

Total: 33

The next series of tests, Test Matrix S1, is designed to study the types and extent of damage caused by impacts to panels with two-ply facesheets. Three different impact energy levels were chosen: 1.4 ft lbs, 1.8 ft lbs, and 3.0 ft lbs. These energy levels are referred to as low, medium, and high, respectively. Two facesheets layups and three core thicknesses are tested in Test Matrix S1. As shown in Table 4.2, a total of 85 specimens are included in this test matrix.

[22]This notation suggested by Lagace. P. A. Lagace, Notch Sensitivity of Graphite/Epoxy Fabric Laminates, Composites Science and Technolgy, Volume 26, 1986, page 95.

Table 4.2 Test Matrix S1

Facesheet Layup	Core Thickness	Impact Energy	Number of Specimens
Compression Specimens:			
(±45)	0.375 in	0.00 ft lbs	3
(±45)	0.375 in	1.40 ft lbs	5
(±45)	0.375 in	1.80 ft lbs	3
(±45)	0.375 in	3.00 ft lbs	5
(±45)	0.687 in	1.40 ft lbs	3
(±45)	0.687 in	1.80 ft lbs	3
(±45)	1.000 in	0.00 ft lbs	3
(±45)	1.000 in	1.40 ft lbs	5
(±45)	1.000 in	1.80 ft lbs	3
(±45)	1.000 in	3.00 ft lbs	5
(0,90)	0.375 in	0.00 ft lbs	3
(0,90)	0.375 in	1.40 ft lbs	5
(0,90)	0.375 in	1.80 ft lbs	3
(0,90)	0.375 in	3.00 ft lbs	5
(0,90)	0.687 in	1.40 ft lbs	3
(0,90)	0.687 in	1.80 ft lbs	3
(0,90)	1.000 in	0.00 ft lbs	3
(0,90)	1.000 in	1.40 ft lbs	5
(0,90)	1.000 in	1.80 ft lbs	3
(0,90)	1.000 in	3.00 ft lbs	5
Tension Specimens:			
(0,90)	1.000 in	0.00 ft lbs	3
(0,90)	1.000 in	1.40 ft lbs	3
(0,90)	1.000 in	3.00 ft lbs	3

Total: 85

Test Matrix S2 is designed to study the effects of static indentations on the same type of specimens as used in Test Matrix S1. The goals of Test Matrix S2 are to study the

force-indentation relation and to compare the damage caused by dynamic impacts and static indentations. Table 4.3 shows Test Matrix S2, which contains 45 specimens.

Table 4.3 Test Matrix S2

Facesheet Layup	Core Thickness	Damage Mode	Equivalent Impact Energy	Number of Specimens
(±45)	0.375 in	static	1.4 ft lbs	3
(±45)	0.375 in	static	3.0 ft lbs	3
(±45)	0.687 in	static	1.4 ft lbs	3
(±45)	0.687 in	static	3.0 ft lbs	3
(±45)	1.000 in	static	1.4 ft lbs	3
(±45)	1.000 in	static	3.0 ft lbs	3
(0,90)	0.375 in	static	1.4 ft lbs	3
(0,90)	0.375 in	static	3.0 ft lbs	3
(0,90)	0.375 in	dynamic	1.4 ft lbs	3
(0,90)	0.375 in	dynamic	3.0 ft lbs	3
(0,90)	0.687 in	static	1.4 ft lbs	3
(0,90)	0.687 in	static	3.0 ft lbs	3
(0,90)	0.687 in	dynamic	1.4 ft lbs	3
(0,90)	0.687 in	dynamic	3.0 ft lbs	3
(0,90)	1.000 in	static	1.4 ft lbs	3
(0,90)	1.000 in	static	3.0 ft lbs	3
(0,90)	1.000 in	dynamic	1.4 ft lbs	3
(0,90)	1.000 in	dynamic	3.0 ft lbs	3
			Total:	45

Three-ply facesheets make up Test Matrix S3. This test matrix contains only a single core thickness and no static indentations. As shown in Figure 4.4, Test Matrix S3 contains 32 specimens.

Table 4.4 Test Matrix S3

Facesheet Layup	Core Thickness	Impact Energy	Number of Specimens
(0,90,-45)	0.687 in	0.00 ft lbs	3
(0,90,-45)	0.687 in	1.40 ft lbs	5
(0,90,-45)	0.687 in	1.80 ft lbs	3
(0,90,-45)	0.687 in	3.00 ft lbs	5
(±45,0)	0.687 in	0.00 ft lbs	3
(±45,0)	0.687 in	1.40 ft lbs	5
(±45,0)	0.687 in	1.80 ft lbs	3
(±45,0)	0.687 in	3.00 ft lbs	5

Total: 32

4.2 Manufacturing Procedures

The manufacture of the test specimens involves nine separate operations: layup, facesheet cure, postcure, trimming, core assembly, panel bond, tab bond, machining, and gauging.

Layup

All facesheets are made from AS4 plain weave graphite fabric impregnated with 3501-6 epoxy. This pre-preg is a net-resin system with a resin content of 34%. The graphite is supplied by two different suppliers: Hercules and Fiberite. Both companies refer to the material as AW193PW/3501-6. The Hercules material may be identified by yellow or red tracer fibers spaced approximately 2 inches

apart. The Fiberite material may be identified by white tracer fibers 6 inches apart. The two materials have very similar properties.

Before cutting, the roll of graphite is removed from the freezer and held at room temperature for 90 minutes before the air-tight bag is removed. The purpose of this is two-fold. First, water vapor tends to condense on the cold roll. By waiting until the roll has reached room temperature before the bag is removed, this problem is avoided. Second, the pre-preg becomes tackier and is easier to work with at room temperature.

The 12 by 14 inch plies are cut from the fabric using a Stanley utility knife with the aid of a 12 by 14 inch Teflon-covered aluminum template. All angles are measured relative to the 0° fibers with the aid of an aluminum templates. Four layups were used: (0,90); (±45); (0,90,0); and (+45,-45,+45). A sheet of Teflon FEP flourocarbon film is then applied to each side of the resulting laminate.

Laminate Cure

The cure layup is shown in Figure 4.5, Cure Layup. Starting from the aluminum cure plate, the layers of material are: non-porous Teflon, the laminate surrounded by the FEP flourocarbon film, a second layer of non-porous Teflon, and then the aluminum top plate. As this is a net-resin system,

no paper bleeder plies are used. Also, as there is no extra resin in the system, it is important that the non-porous Teflon be wrapped tightly around the laminate and sealed with flash tape. The entire assembly in covered with a fiberglass cloth and then enclosed in a vacuum bag.

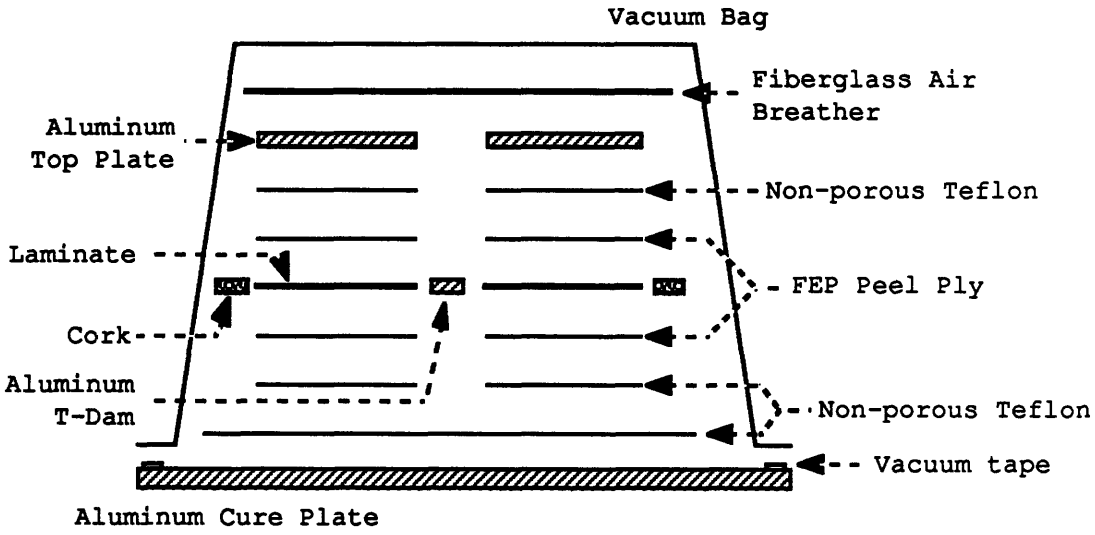


Figure 4.5 Laminate Cure Layup

The laminates are cured under a 135 psi pressure differential (15 psi vacuum and 120 psig autoclave pressure) using a two-step process. The first step is a flow stage at 225°F. This temperature is chosen because it is the minimum viscosity temperature of the epoxy. The flow temperature is maintained for 60 minutes to allow the epoxy to evenly distribute itself throughout the laminate. After the flow stage, the temperature is raised to 350°F to allow the epoxy to cure. The cure temperature is maintained for 120 minutes. All temperature changes take place at the rate of 5°F per minute to prevent thermal shock. Time histories of the cure

temperature, pressure, and vacuum are shown in Figure 4.6-4.8 respectively.

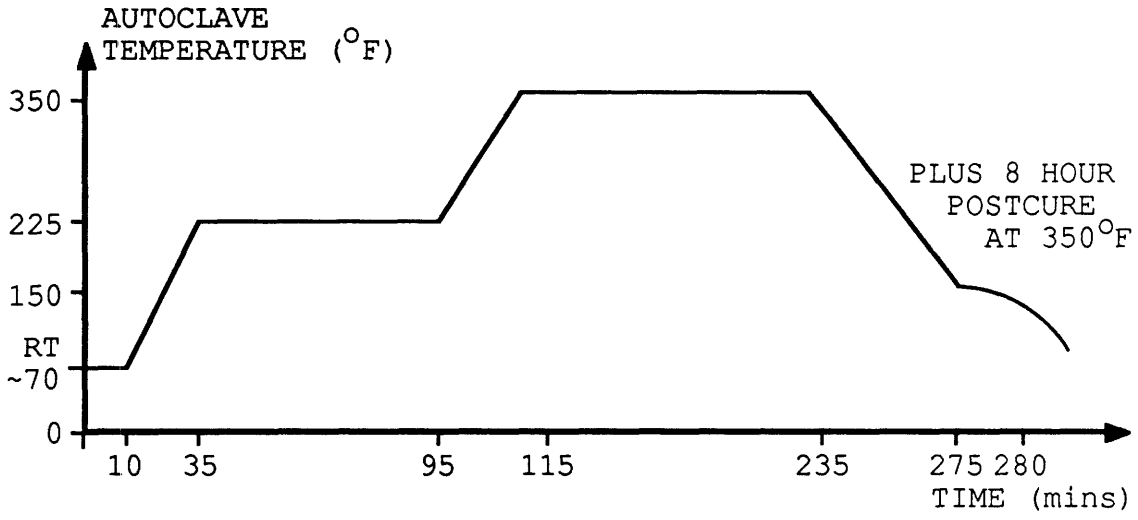


Figure 4.6 Laminate Cure Temperature History

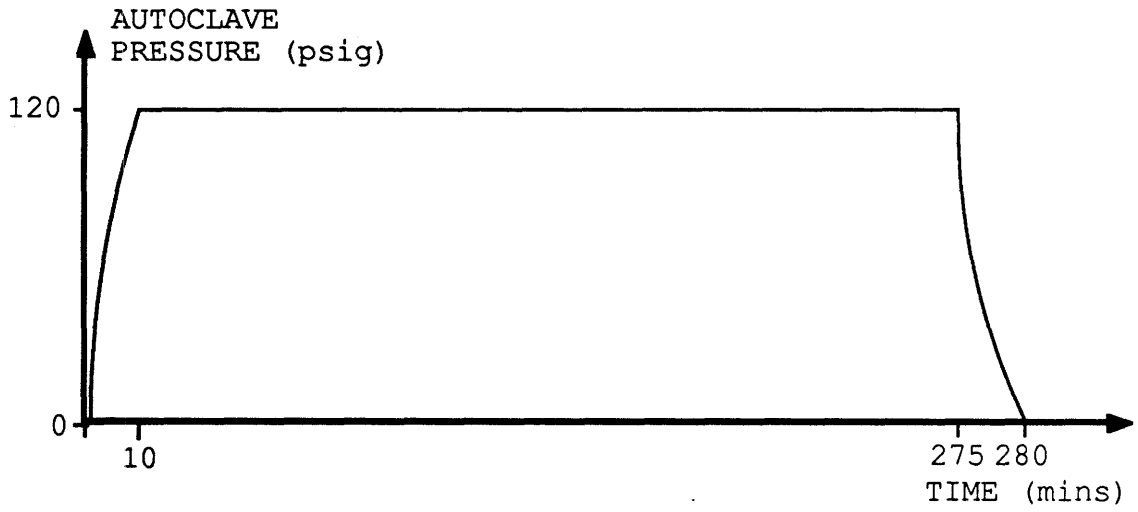


Figure 4.7 Laminate Cure Pressure History

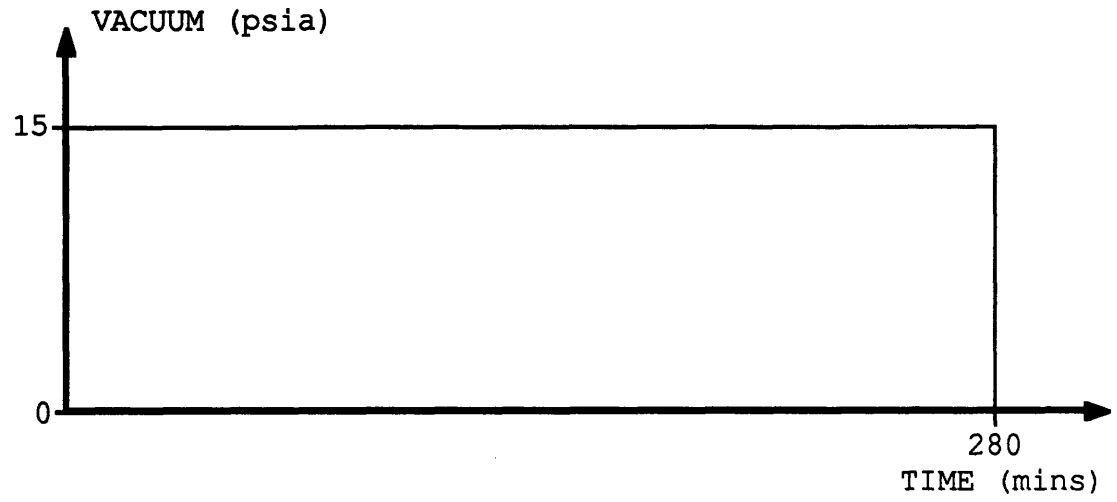


Figure 4.8 Laminate Cure Vacuum History

Postcure

Following the cure, the laminates are removed from the cure assembly and postcured for 8 hours at 350°F. No pressure or vacuum is applied to the laminates during the postcure.

Trimming

The next step in the manufacturing procedure is to trim the edges of the laminates. This is done to remove any epoxy ridges which might prevent the facesheet from lying flat against the honeycomb core. Cutting is done with a diamond grit cutting wheel mounted on a milling machine. The 5-inch diameter wheel rotates at 1100 rpm and the cutting table is set to advance at a rate of 11 inches per minute. A low-

velocity water-jet is used to prevent heat buildup and to wash away cutting residues.

Core Assembly

The cores of the specimens are Nomex honeycomb with a nominal density of 3.0 pcf. This material does not have sufficient stiffness to support the grips of the testing machine. Therefore, aluminum honeycomb with a density of 22.0 pcf is substituted for the Nomex for 2.5 inches at each end of the specimen. The aluminum honeycomb sections are bonded to the Nomex using Hysol Clear Epoxy-Patch hand-mixed epoxy with the aid of an assembly jig. This epoxy cures in 8 hours at room temperature.

Bond Cure

The facesheets are bonded to the cores with FM-123-2 film adhesive made by American Cyanimid. The density of the film adhesive is 0.06 lbs/sq ft. Non-porous Teflon sheets are placed on each side of the panels. Next, steel top plates are used to completely cover the panels. An aluminum bar is placed along the side of the panels to prevent the vacuum bag from damaging the Nomex cores. Fiberglass air-breather is placed on top of the top plates and the entire assembly is enclosed in a vacuum bag. The panel bond cure arrangement is shown in Figure 4.9.

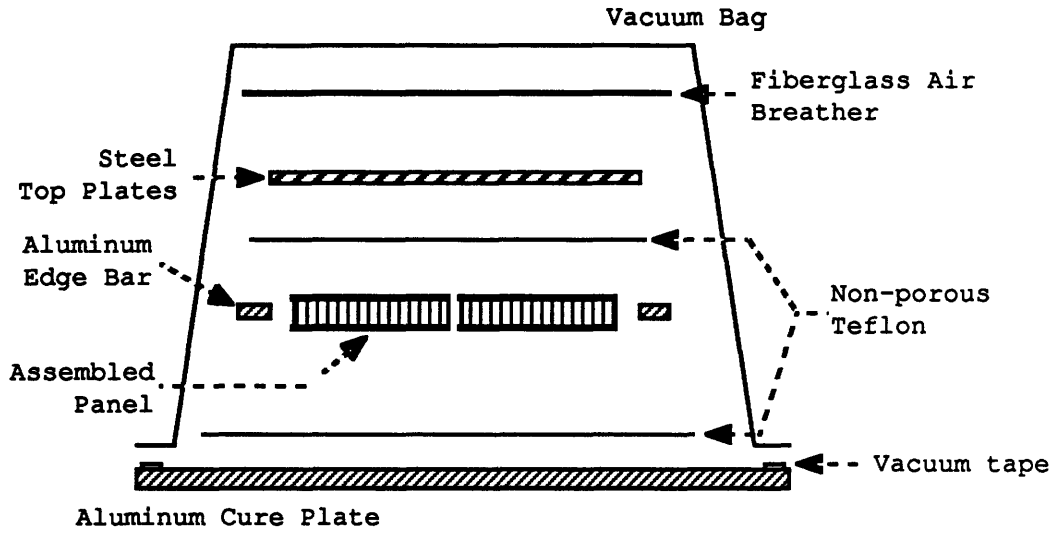


Figure 4.9 Panel Cure Assembly

The vacuum bag was left vented to atmospheric pressure to prevent damage to the Nomex cores. An autoclave pressure of 40 psi is used. The bond cure cycle consists of a single stage at 225°F for 120 minutes. Time histories of temperature and pressure for the bond cure are shown in Figures 4.10 and 4.11 respectively.

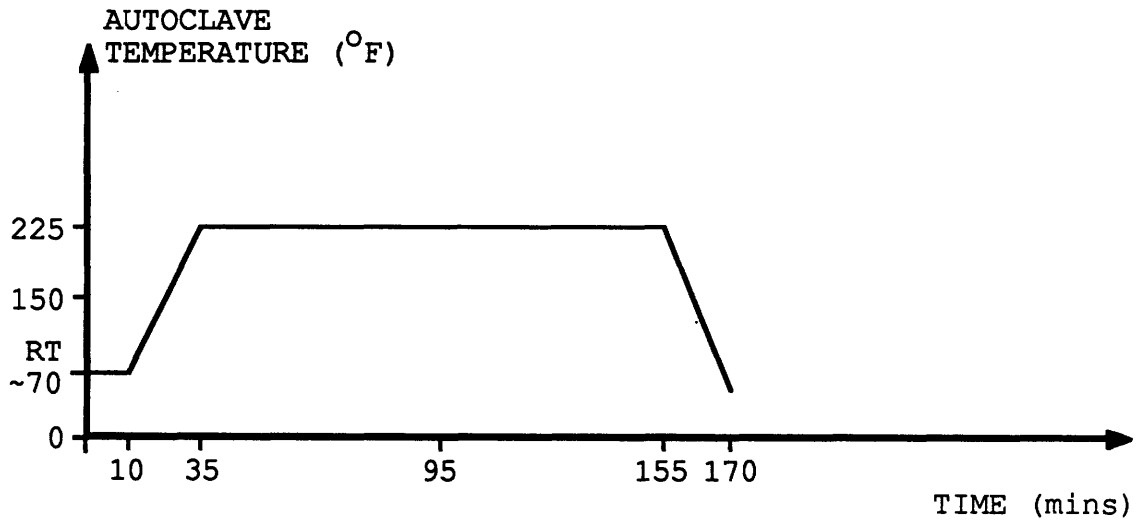


Figure 4.10 Panel Cure Temperature History

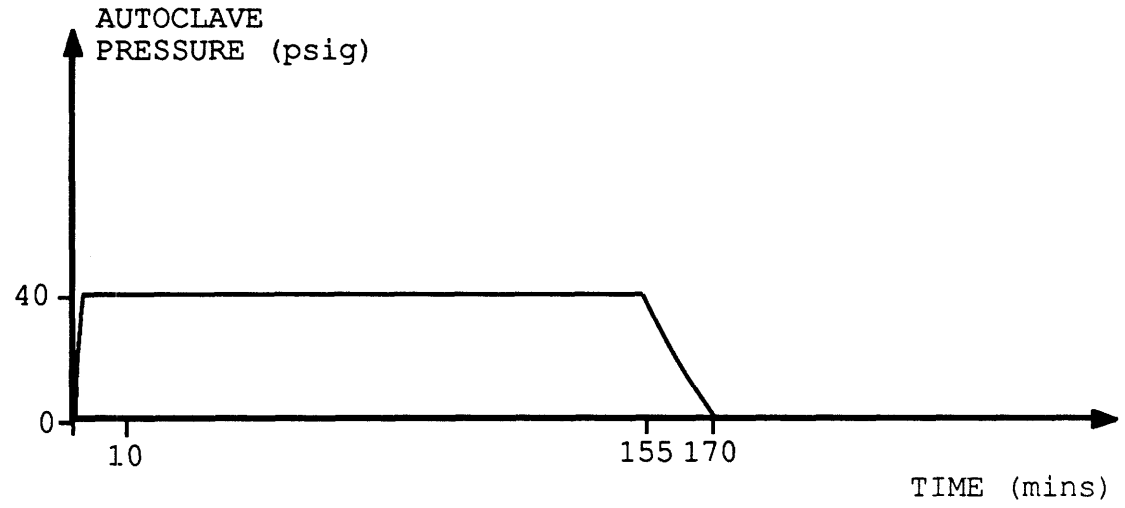


Figure 4.11 Panel Cure Pressure History

Tab Cure

Pre-cured fiberglass loading tabs are used to distribute load into the specimens and thus to prevent the testing machine grips from inducing failure. These tabs are 3M Scotchply Type 1002 Crossply. The tabs used are 7 plies thick with a nominal thickness of 0.07 inches. They are bonded to the panels with the same adhesive and procedure used to bond the facesheets to the cores.

Machining

After the tab cure, each 12 wide panel is cut into three 3.5 inch wide specimens. Again, the cutting is done with a water-cooled diamond grit cutting wheel. For cutting panels, an 11 inch wheel is used which rotates at 700 rpm. The table

feed rate is 5 inches per minute. The finished specimens have nominal dimensions of 3.5 by 14 inches. Figure 4.12, Test Specimens, shows the configuration of the specimens.

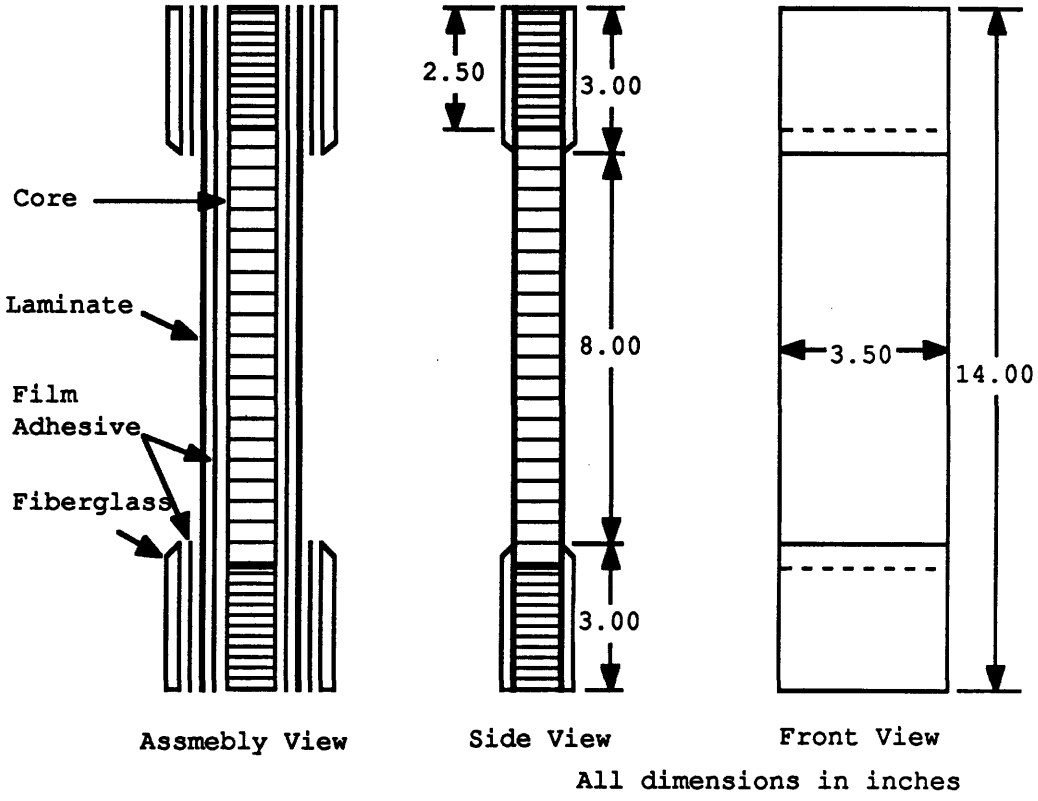


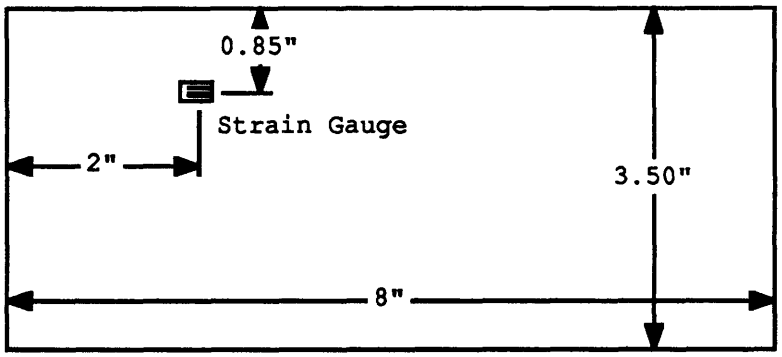
Figure 4.12 Test Specimens

Measuring

Measurements of the specimen width are taken using a caliper at each end and the middle of both facesheets of each specimens. The resulting six values are averaged to find the width of the specimen.

Gauging

The last step in the manufacturing process is to apply strain gauges to the specimens. A Micro-Measurements type EA-06-125AD-120 strain gauge is applied to each facesheet. The gauges are bonded with a cyanoacrylate adhesive according to the gauge manufacturer's specifications. The location of these gauges is shown in Figure 4.13, Strain Gauge Locations.



Note: One gauge is placed on each side of the panel in a back-to-back arrangement.

Figure 4.13 Strain Gauge Locations

4.3 Testing Procedure

Four different types of tests are performed: beam bending, dynamic impact tests, static indentation tests, and residual strength tests. If a specimen is to be damaged, it is subjected to a dynamic impact or a static indentation, never both. Following the impact or indentation, an x-ray

inspection of the damaged specimen is made. The last step is to subject the specimen to a residual strength test.

Beam Bending

Several specimens of various configurations are subjected to beam bending tests in order to determine the bending regidity of the facesheets and film adhesive. One end of the specimen was clamped to a rigid support and weights were attached to the free end. The tip displacement was then measured with the aid of a ruler. A mirror was used to prevent parallax.

Dynamic Impact Tests

Specimens to be impacted are mounted in a holding jig which is designed to provide clamped boundary conditions along the short edges while leaving the long edges free. Figure 4.14 depicts the holding jig.

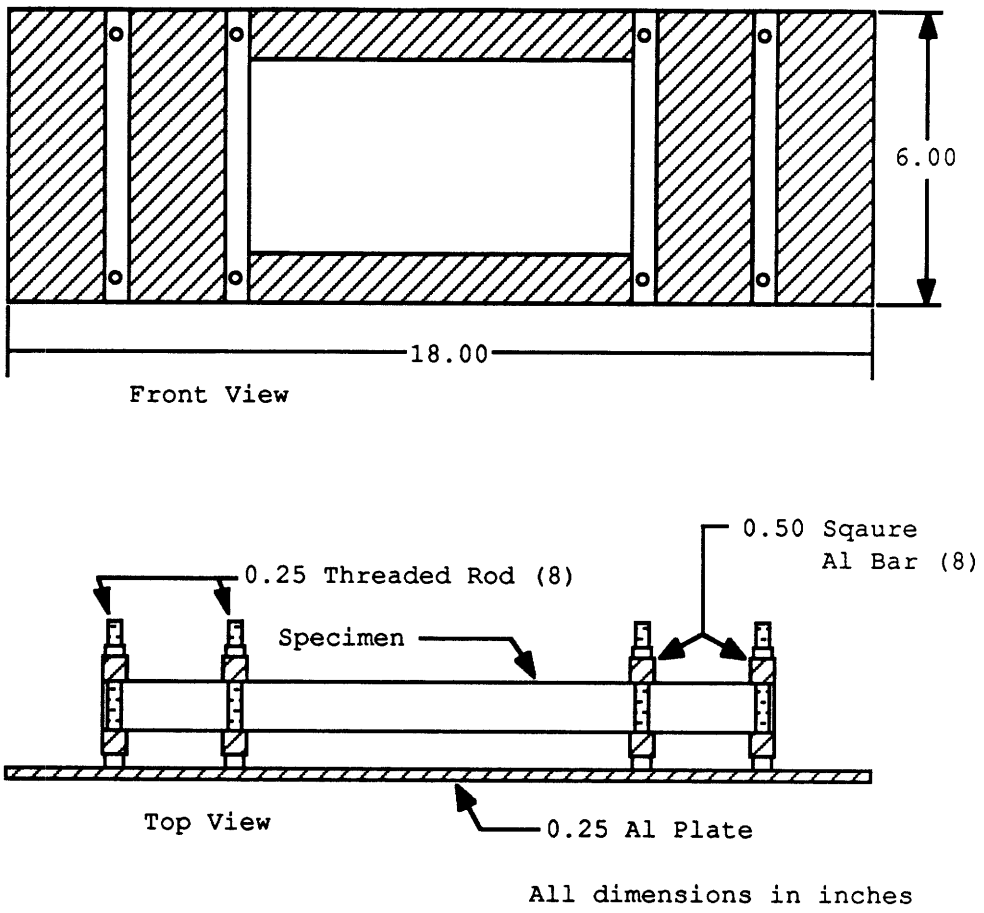


Figure 4.14 Specimen Holding Jig

Four sets of 1/2 inch square cross-section aluminum bars are used to secure the specimen and provide the clamped boundary condition. The jig itself is mounted to a rigid steel frame. The frame supports the entire back surface of the jig, so that the jig is not subjected to bending.

The impactor is a 26 inch steel rod mounted on linear bearings so that it is free to travel along its axis. Attached to the rod are a PCB Model 208A05 Force Transducer and a 1/2 inch diameter, hemi-spherical, stainless-steel impact head known as a tup. Around the center of the rod is

a 1/2 inch thick plastic "doughnut" which is used as a timing flag to interrupt a light beam. A CENCO Model 31709 timing system times the length of interruption to determine the velocity of the impactor. The mass of the impactor and attachments is 0.105 slugs. Figure 4.15 is a drawing of the impactor with attachments.

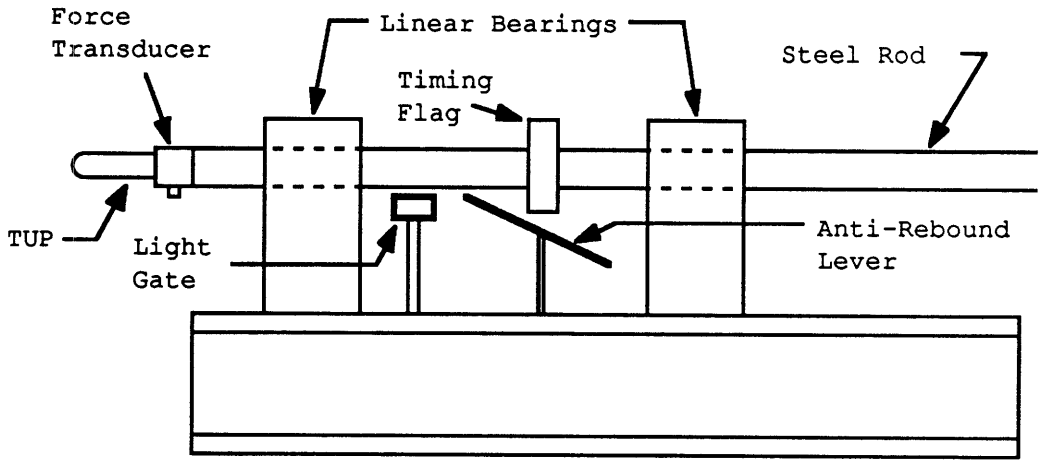


Figure 4.15 Impactor

Multiple impacts are prevented by the anti-rebound lever located to the right of the light gate. Figure 4.16 is a close-up view of the lever.

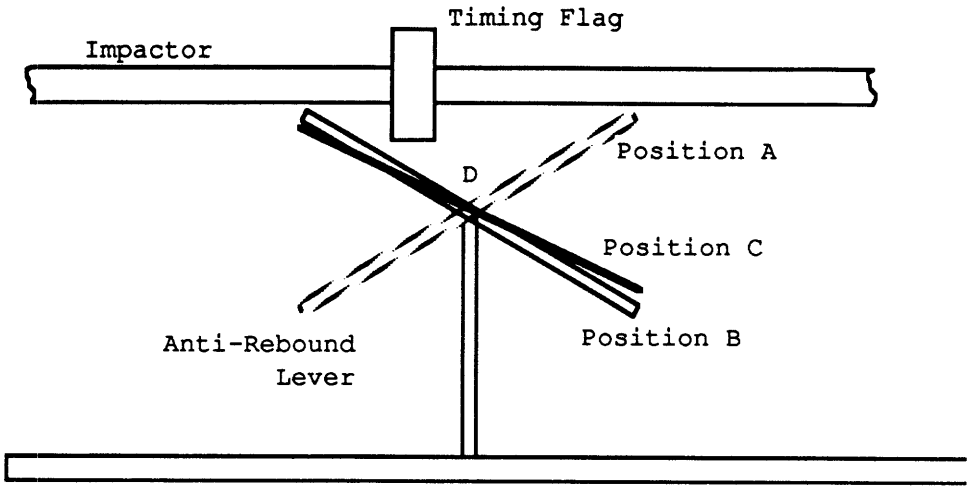


Figure 4.16 Anti-Rebound Lever

The lever rotates around pivot D and has a range of motion from position A to position B. It is spring loaded to remain in either position A or position B. The neutral position is labeled C. If the lever is displaced clockwise form position C it will snap into position B. Similarly, if the lever is displaced counter-clockwise from position C, it will snap into position A.

Before impact, the lever is placed into position B. As the impactor slides forward, the timing flag hits the lever which rotates counter-clockwise past position C. The lever then snaps into position A. After the impact, the impactor slides backwards and the timing flag hits the opposite end of the lever. The lever rotates clockwise, but never reaches position C. When the impactor passes, the lever returns to postion A. Continuing its rebound, the impactor often hits its stop and bounces back towards the specimen. However, the anti-rebound lever is now in position A and does not allow

the impactor to pass. The anti-rebound lever is 100% effective in preventing multiple impacts.

The impactor is set in motion by a spring-loaded striker mechanism. The striker is shown in Figure 4.17. The entire device including stiker and impactor is known as FRED. This name has no significance. The winch at the end of the unit is used to compress the main spring at the other end. A ruler is mounted to FRED to measures the compression of the spring. The striker is coupled to the winch cable with two 1500 lb capacity electromagnets. To initiate an impact, the electromagnets are de-energized.

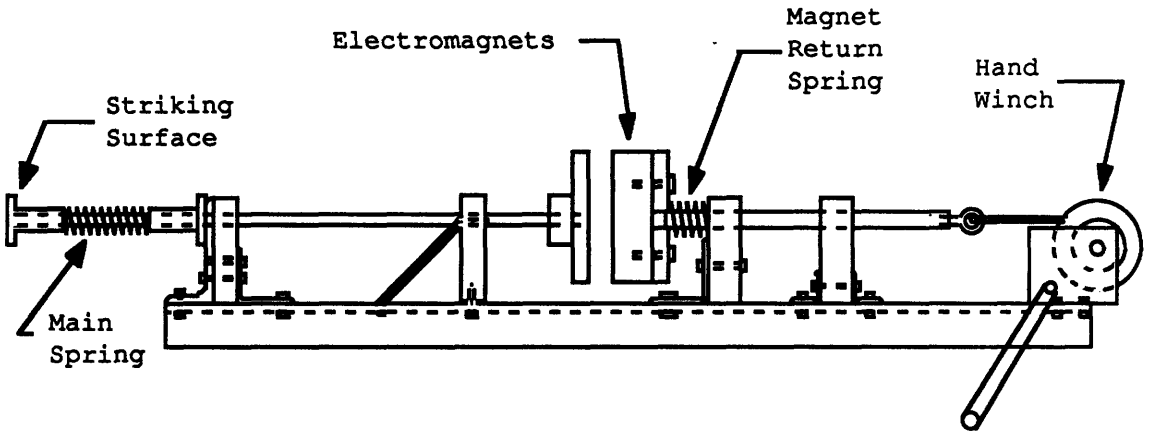


Figure 4.17 FRED - Striker Unit

Data is collected by a DEC Micro PDP-11/23 computer equipped with a Data Translation DT-3382-G-32DI analog-to-digital converter. The signal from the force transducer is sampled at a rate of 25 kHz and this data is stored for later analysis. Data collection is triggered by the falling edge

of the signal from the CENCO timing unit. Figure 4.18 is a schematic of the impact test data collection system.

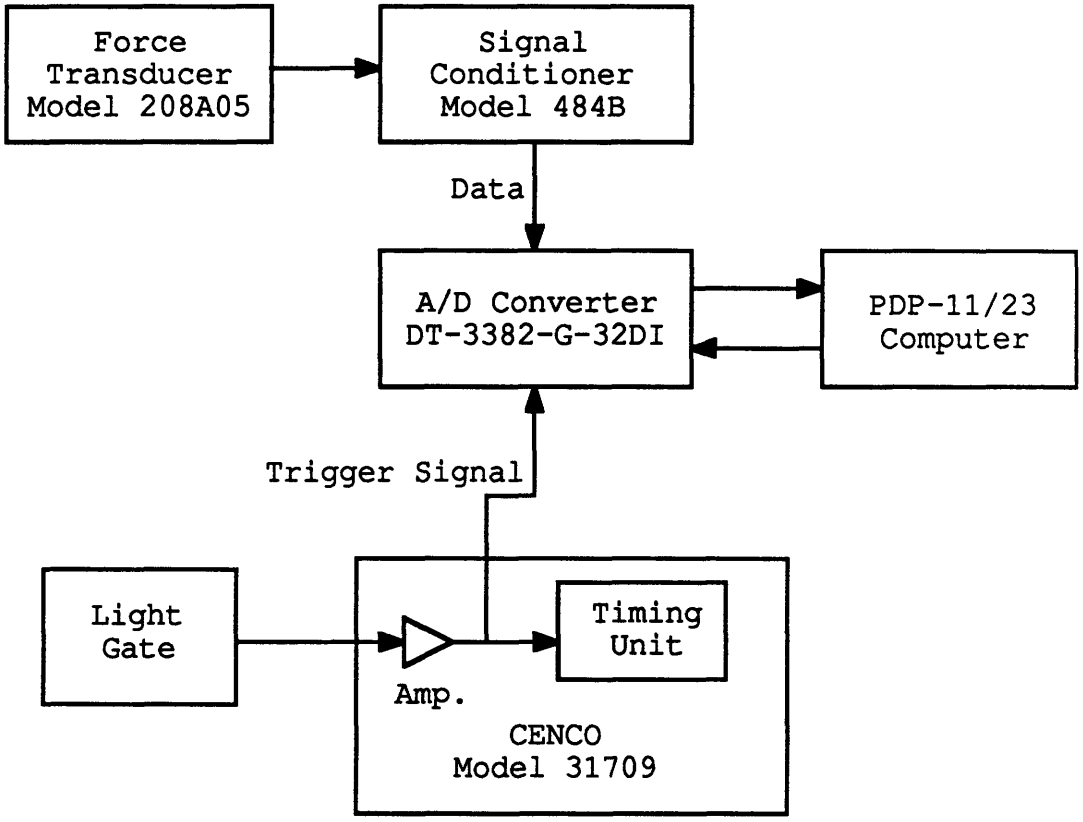


Figure 4.18 Impact Test Data Collection System

Static Indentations

Static Indentations are performed using an MTS-810 uni-axial testing machine. The same impact head and force transducer are used for both the dynamic impact and static indentation tests. The holding jig described earlier is not used. Instead, the specimen to be statically indented is placed directly upon a rigid support. Figure 4.19 depicts the Static Indentation Apparatus.

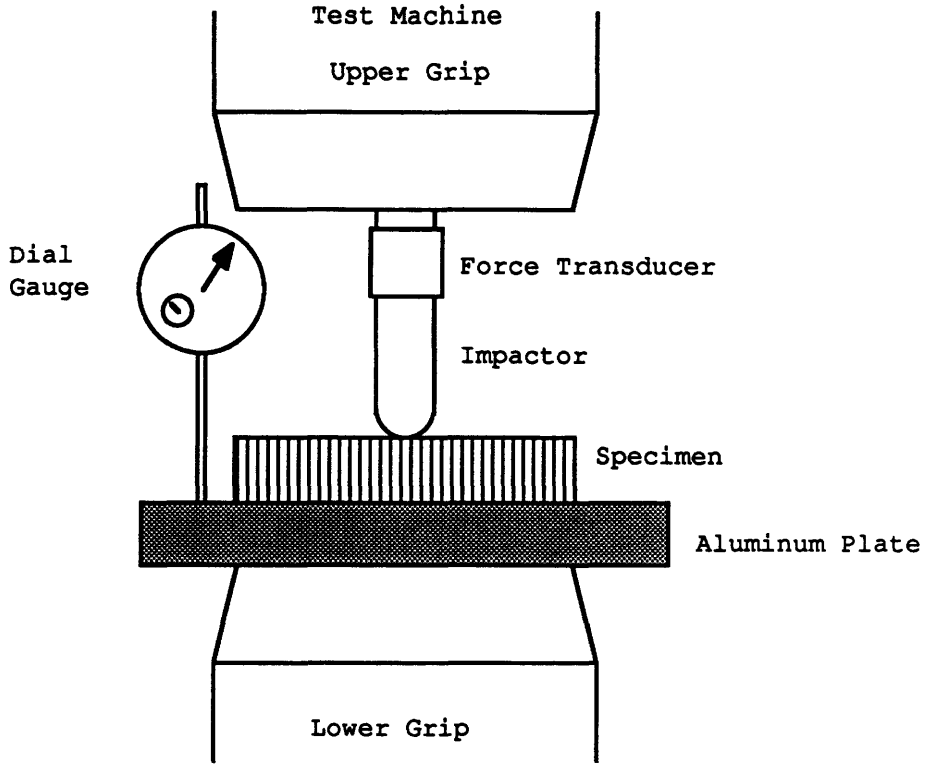


Figure 4.19 Static Indentation Apparatus

The test machine holds the impact head fixed and drives the specimen upwards at a constant rate of 0.002 inches per second. As shown, a dial gauge is used to monitor the total displacement. When the indentation reaches the desired depth, the stroke direction is reversed and the specimen returns to its original position. Data is collected from the force transducer and the stroke transducer of the test machine by a DEC PDP-11/34 computer and stored for later analysis.

X-Ray Inspection

Impacted and indented specimens are subjected to x-ray inspection. The Scanray Torrex 150D X-Ray Inspection System is set to operate at a differential of 50 kVolts. An x-ray opaque dye, 1,4-Diiodobutane, is applied to the damaged region with a cotton-tipped swab applicator. DIB is added until the specimen no longer absorbs the dye. The specimen is then placed damaged side down on a sheet of Polaroid Type 52 Polapan 4x5 Instant Film inside the x-ray chamber. The x-ray machine is set to expose the film to 240 mrad using the "TIMERAD" control. After exposure, the film is processed according to the manufacturer's specifications. This method produces a full size image of the damage area.

Specimen Sectioning

A few specimens are sectioned through the impact site after impact. Specimens are sectioned with an 11-inch cutting wheel rotating at 700 rpm. Table feed speed is set to 5 inches per minute. The exposed surface is then examined visually with the naked eye and with the aid of a microscope.

Residual Strength Testing

The residual strength tests are conducted on the MTS-810 uniaxial test machine. Textured, flat grips are installed in the test machine and the grip pressure is set to 500 psi.

The specimen is aligned so that the load direction is parallel to the long sides and gripped in the upper grip. The strain gauges are then zeroed and calibrated. Next, the lower grip is closed. At this point, the strain gauges are monitored. If the extensional strain in the two facesheets differ by more than 200 μstrain, the specimen is ungripped from the lower grip and masking tape is used to shim the specimen. This process is repeated until the strain in the facesheets are within 200 μstrain of each other.

The test machine then begins to compress the specimen at a constant stroke rate of 0.002 inches per second. The test continues until the specimen fails. Failure is defined as a full width fracture of one facesheet or general structural instability caused by separation of a facesheet from the core. Data is collected from the two strain gauges and from the load and stroke transducers on the test machine. The failure load is also read directly from the memory of the test machine.

5. Results

5.1 Global Model Performance

In any assumed modes analysis, it is important to determine the number of modes required to achieve a converged solution. The number of modes required for convergence depends upon the relative values of the Hertzian spring constant, k, and the plate stiffness, D[23]. Impact problems of this kind can be thought of as two coupled dynamic systems: the one degree-of-freedom Hertzian spring, and the multi-degree-of-freedom plate theory. If k is small compared to D (a soft spring and stiff plate), then the Hertzian spring is compressed while the plate remains relatively unaffected. This is, in essence, a one degree-of-freedom system. On the other hand, if k is large compared to D, then the spring is largely unaffected and the plate is deformed. In the limit as k/D approaches zero, the model becomes a one degree-of-freedom system and no plate modes are required for an exact solution (to the approximate problem).

Figure 5.1 is a graph of force-time histories generated by the global model "SPAD" using various numbers of modes. For the specimens and impacts considered, the Hertzian spring

[23]Unpublished idea proposed by P. H. Wilson Tsang, Doctoral Candidate, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1988.

is fairly soft compared to the plate bending stiffness and the graph shows that 2x2 modes are required to obtain a converged solution.

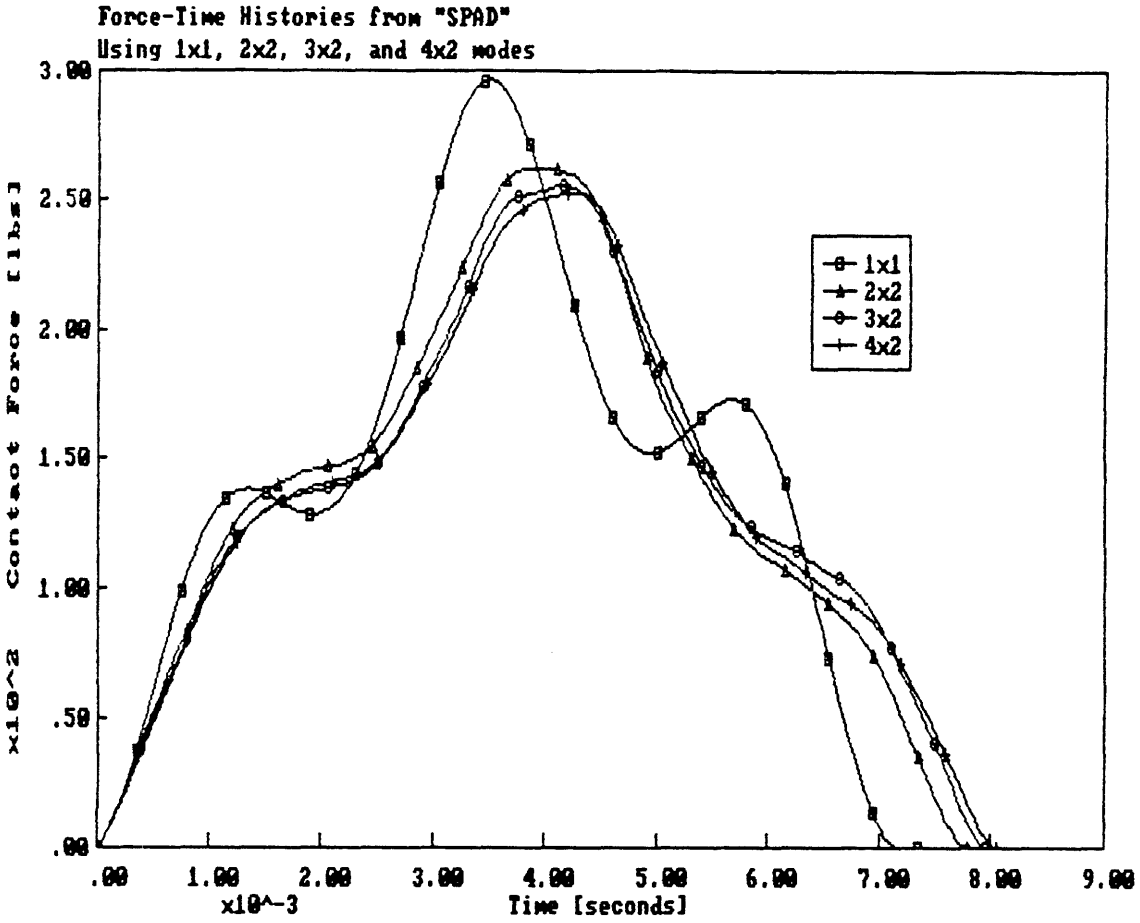


Figure 5.1 Global Model Convergence (Modes)

Another important parameter is the required length of the time step in the numerical integration scheme. Figure 5.2 is a graph of force-time histories obtained using different time steps. This graph shows that a time step of 100 μseconds is sufficient to obtain a converged solution.

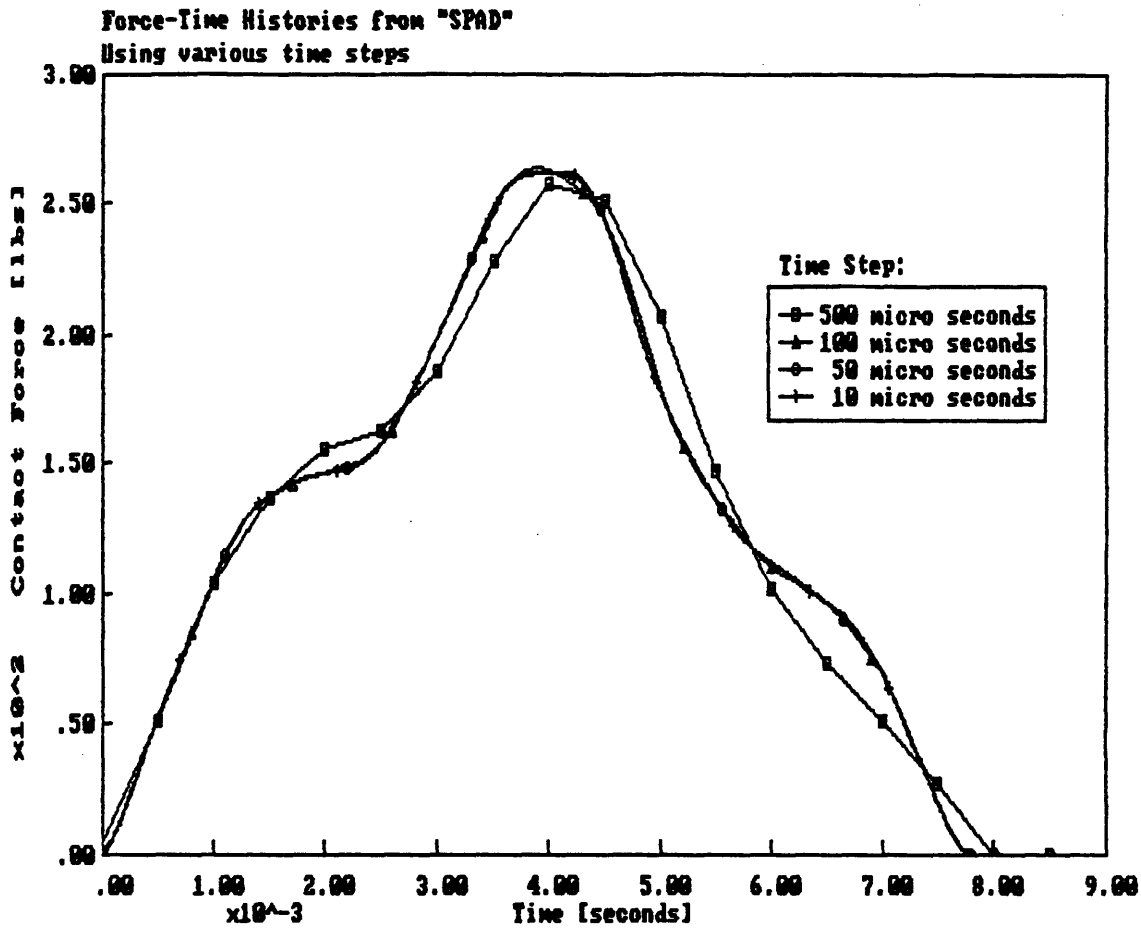


Figure 5.2 Global Model Convergence (Time Step)

5.2 Local Model Performance

The validity of the local model can be checked in several ways. The ultimate test is, of course, comparison with experimental data. However, it is possible to evaluate the local model for consistency. One such check is the total elastic strain energy. From the global model, a value of α_{max} is determined. The strain energy stored in the panel is equivalent to the external work done by the impactor and is given by:

$$W_{ext} = \frac{2}{5} k \alpha_{max}^{5/2} \tag{79}$$

The strain energy is calculated by the local model and can be compared to the external work calculated using equation 79. This may seem like a circular argument because the local model is used to determine the value of the constant k. However, because the analysis forces the parameter β to be equal to 3/2, it is necessary to check the resulting strain energy. Figure 5.3 is a graph of the external work, W_{ext}, and the total elastic strain energy, Π as a function of the impact energy. Ideally, these curves should be identical.

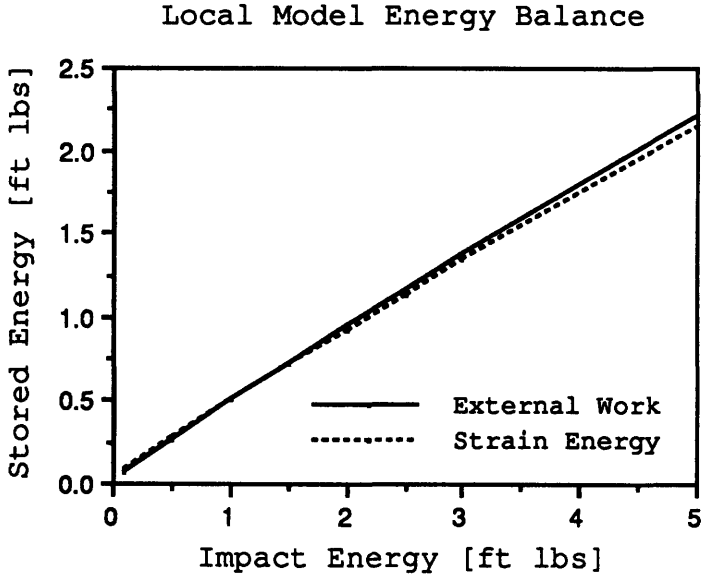


Figure 5.3 Local Model Energy Balance

5.3 Material Properties

Experimental values were determined for modulus, Poisson's ratio, tensile strength, and failure strain of the

facesheets; buckling load for the sandwich panels, and an effective modulus for the stiffening effect of the core and embedded film adhesive. These values are shown in Table 5.4, Material Properties.

Table 5.4 Material Properties

Facesheets: AW193PW/3501-6

Longitudinal Modulus E_L: (warp direction)	9.30 msi
Transverse Modulus E_T: (fill direction)	8.81 msi
Poisson's Ratio ν_{LT}:	0.09
Shear Modulus G_{LT}:	0.70 msi
Tensile Strength F_{TU}:	116.1 ksi
Failure Strain ε_{ult}:	12000 μstrain

Sandwich Panels: (0.687 inch core)

Buckling Loads P_{cr}:	
(0,90) facesheet:	45.0 ksi
(±45) facesheet:	29.4 ksi

Film Adhesive: (including embedded Nomex)

Effective modulus:	1.20 msi

5.4 Damage Types - Two-Ply Facesheets

The following types of damage are observed: core crushing, matrix cracking, fiber breakage, and delamination. At low energy levels, the first damage to occur is core crushing. Examination of sectioned specimens reveals that the core crushing is buckling of the individual cell walls of the Nomex material. Core crushing begins to occur at 0.7 ft

lbs of impact energy. Often, the core will be damaged while the facesheets are intact. This type of damage will not appear during an x-ray inspection but can be found by sectioning the specimen. Figure 5.5 depicts this type of damage.

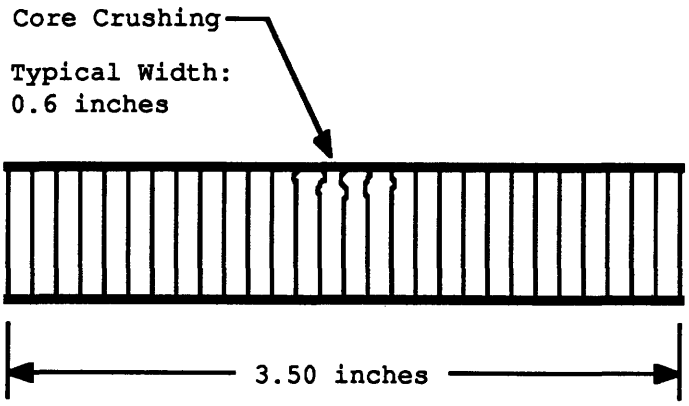


Figure 5.5 Core Crushing

At 1.0 ft lbs, the facesheets begin to exhibit damage. The first external indication of damage is a "scuff mark" which is indicative of very localized matrix damage. Concurrently, matrix cracks begin to form between the tows in the woven facesheets.

At 1.4 ft lbs, core crushing becomes more evident and a noticeable dimple remains in the panel. Matrix cracking occurs throughout the dimple and is most evident at the shoulder of the dimple. Fiber breakage begins to appear at tow boundaries within the dimple.

At 3.0 ft lbs, extensive fiber breakage occurs within the dimple. Little damage is seen outside the dimple. The pattern of fiber breakage is a cross aligned with the tows. This results in four triangular flaps which separate under the force of the impactor. Figure 5.6 represents the pattern of fiber breakage.

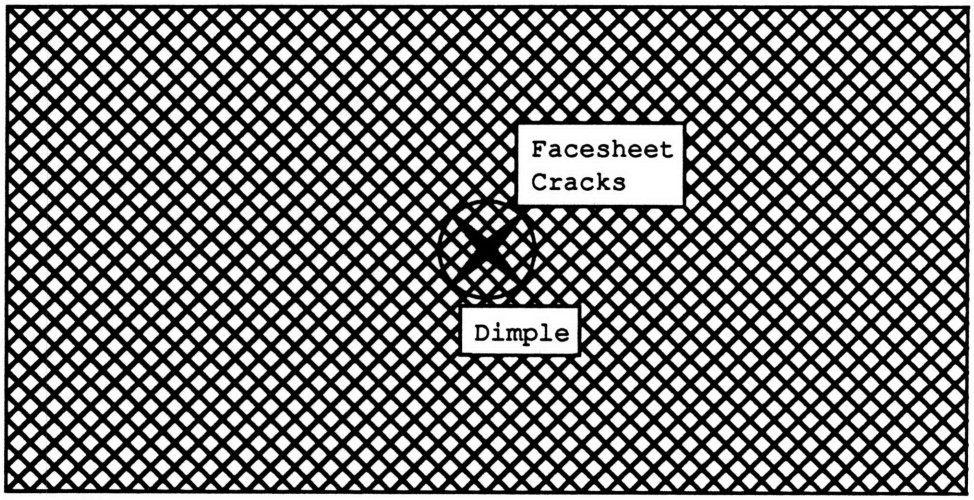


Figure 5.6 Fiber Breakage Pattern

Note: The diagonal lines represent tows.

X-ray inspection reveals that delaminations occur with fiber breakage in the dimple. This type of extensive damage is accompanied by tensile fracture of the film adhesive leading to areas of debonding between the facesheet and core.

5.5 Damage Types - Three-Ply Facesheet Specimens

Three-ply facesheet specimens exhibit the same types of damage as two-ply specimens with the following exception: delaminations are found in low energy level impacts. Unlike

the two-ply specimens, these delaminations often occur before visible facesheet damage. The delaminated area is usually about 0.6 inches in diameter. There are two reasons for the early appearance of the delaminations. First, the three-ply facesheets are thick enough to develop local bending stresses within the facesheet. Second and more importantly, the three-ply facesheets contain an interface of material property mismatch. This interface is between the 90° and -45° plies in the (0,90,-45) and between the -45° and 0° plies in the (±45,0). In a plain weave fabric, a 0° and a 90° ply are identical, as are a +45° and a -45°. No such material mismatch occurs in the two-ply specimens.

5.6 Static Indentation Damage

Static indentations produce the same types of damage as dynamic impacts. Core crushing, facesheet cracking, fiber breakage, and delaminations present in dynamic impacts are also seen in static indentations. It was not possible to differentiate damage caused by dynamic impacts and static indentations.

5.7 Force-Time History

Figure 5.7 is a graph of the experimental and analytical force-time histories of a 1.32 ft lb impact in which no visible damage occurred. The experimental data is represented by the solid line and the analytical prediction

is represented by the dashed line. Note that the contact force increases up to a maximum value and decreases back to zero. This graph is a typical example for specimens in which no fiber breakage was observed and shows reasonable agreement with the analytical curve.

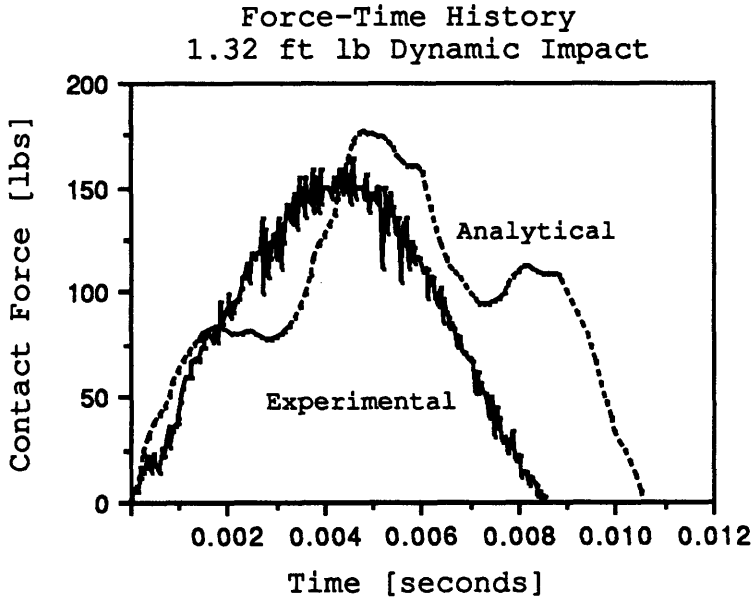


Figure 5.7 Force-Time History - 1.32 ft lb Impact

Figure 5.8 is a graph of the experimental and analytical force-time history of a 1.49 ft lb impact in which fiber breakage was observed. In this case, the measured force increases smoothly up to a maximum, but then a sudden drop occurs. Accompanying the drop is the onset of oscillations in the contact force. This graph is a typical example for specimens in which fiber breakage was observed. Note that the experimental and analytical curves separate at this point. This separation occurs because the analytical model does not account for the occurrence of damage.

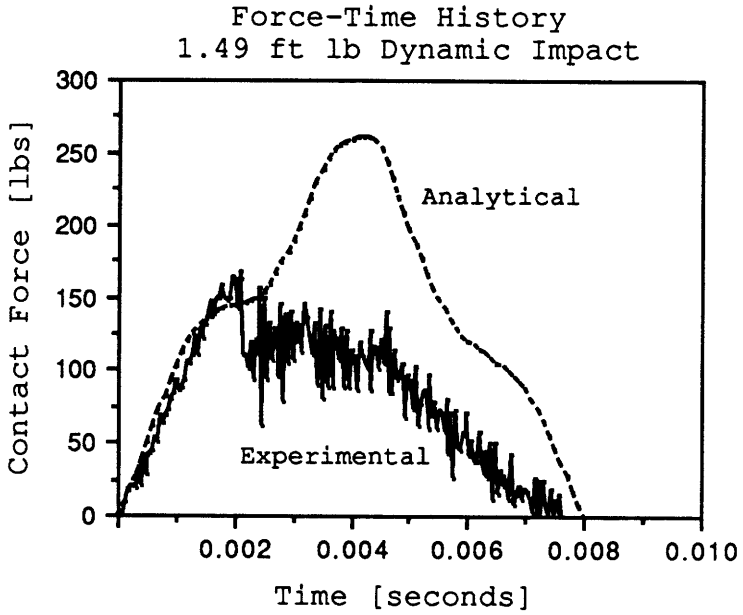


Figure 5.8 Force-Time History - 1.49 ft lb Impact

The sudden drop in load seen in Figure 5.8 at 0.002 seconds is caused by cracking of the facesheet. As the strains induced by the impactor reach some critical value, damage occurs, and the contact force immediately drops. The damage consists of fiber breakage along a tow boundary in either the warp or the fill direction. Fibers at 90° to the broken fibers are parted. The result is a crack though both the warp and fill fibers running along a tow boundary. This type of behavior would be expected to initiate vibrations in the facesheet and impactor. These vibrations would manifest themselves as oscillations in the contact force, as is evidenced.

5.8 Force-Indentation Relation

From each static indentation test, contact force and indentation data is collected. This data is then fit to the two-parameter function $F = k\alpha^{\beta}$ (see Chapter 2, equation 61). This method can also be applied to the data generated by the local model. The results of these analyses are shown in Table 5.9 The value of k is strongly dependent on the value of β, thus no comparison can be made among the values of k determined for different values of β.

Table 5.9 Experimental Curve-Fit Parameters

Parameter	Experimental Average Value	Analytical Value
k:		7295 lbs $in^{-1.34}$
β:	1.36 (CV=12.3%)	1.34

Figure 5.10 shows force-indentations data from two experiments as well as an analytical prediction. Note that the analytical model predicts more force than is measured experimentally. This implies that the panel model is too stiff, as would be expected from any assumed modes Rayleigh-Ritz solution.

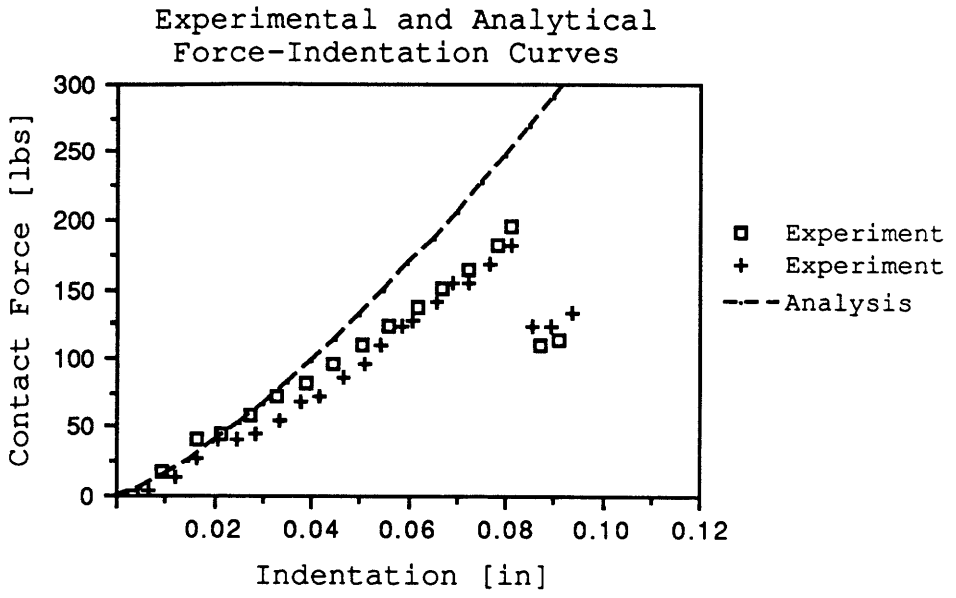


Figure 5.10 Experimental and Analytical Force-Indentation Curves

Note: The sudden drop in contact force at 0.085 inches is explained in the discussion of Figure 5.12.

If the exponent β is constrained to the theoretical value of 3/2, the value of k which best fits the analytical data is 12025 lbs $in^{-3/2}$. The difference between this value and the previously determined value of 7295 lbs $in^{-1.34}$ is a result of the sensitivity of k to changes in the exponent β.

Figure 5.11 is a graph of data collected during a 0.09 inch static indentation test in which no fiber breakage was observed. Superimposed on this data is the analytical curve obtained from the one-parameter fit with β constrained to be 3/2.

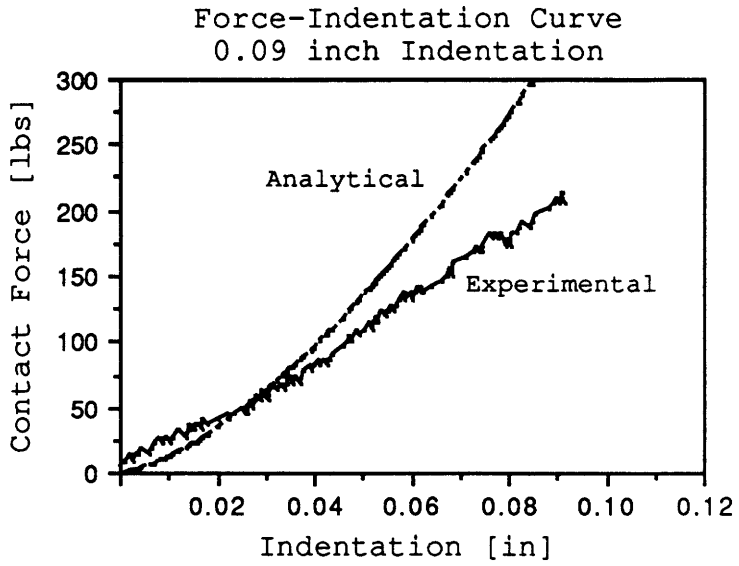


Figure 5.11 Force-Indentation Curve 0.09 in Indentation

Figure 5.12 is a graph of data collected during a 0.12 inch static indentation test in which fiber breakage was observed. The analytical curve is the same as found in Figure 5.11.

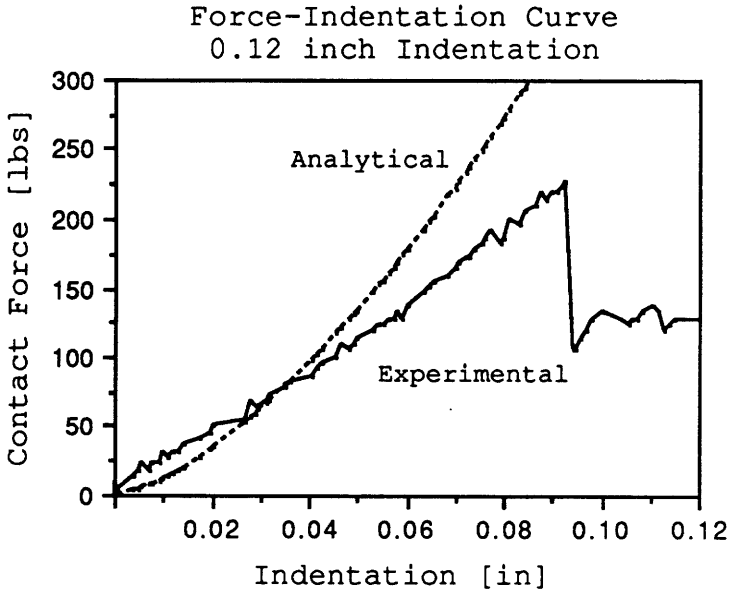


Figure 5.12 Force-Indentation Curve 0.12 in Indentation

In this indentation, a sudden drop in load is encountered at an indentation of 0.095 inches. Similar drops may be seen in the experimental data in Figure 5.10. The drop is accompanied by a loud cracking sound and represents the point of fiber breakage. No drop appears in the analytical curve because the local model does not account for the occurrence of damage. Note the similarity of Figure 5.12 to the force-time history shown in Figure 5.8. Both cases show a smooth rise in contact force up to a maximum followed by a sudden drop. While vibrations are detected in the dynamic impact test, no vibrations are detected in the static indentation test due to the slow data collection rate (2 Hz).

5.9 Damage Quantitation

Figure 5.13 is an x-ray of a damaged specimen. The honeycomb structure of the core can easily be seen. The darker area is the impact site. The dark grey lines represent facesheet cracks while the more diffuse, lighter grey region is the area of delamination.

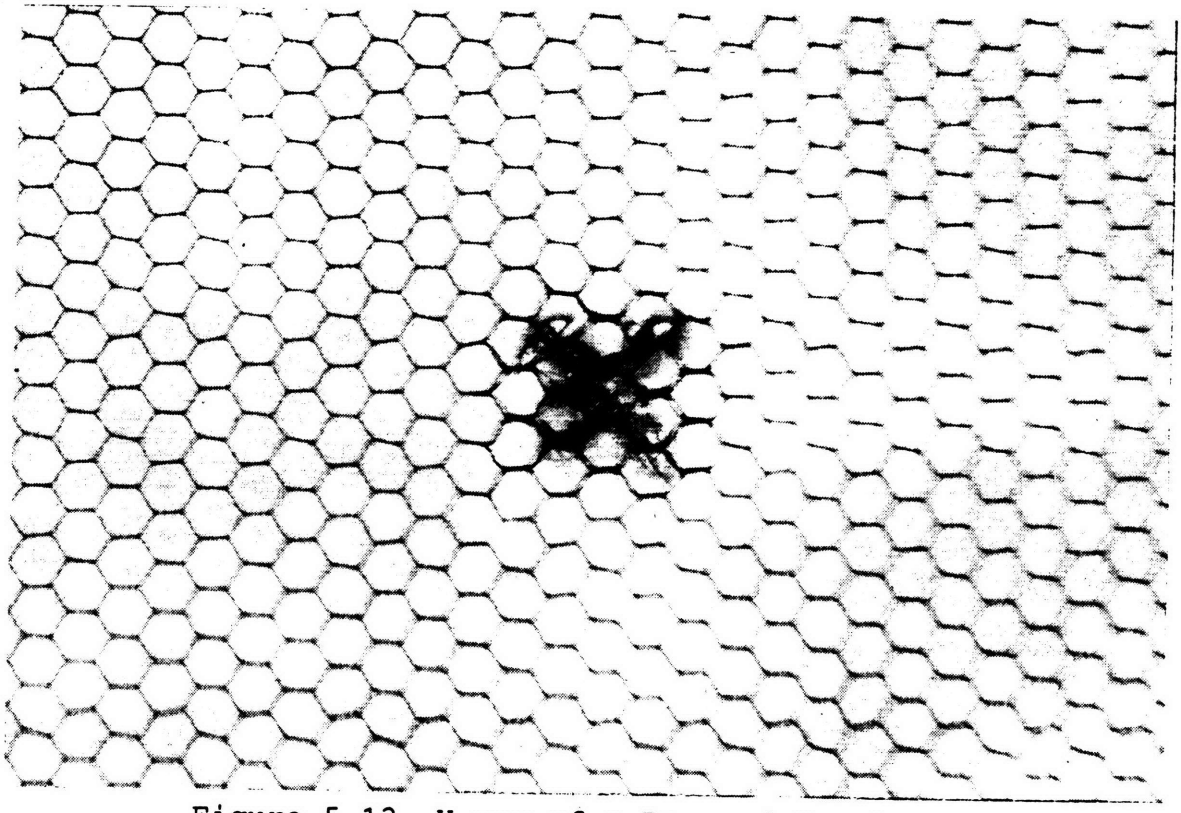
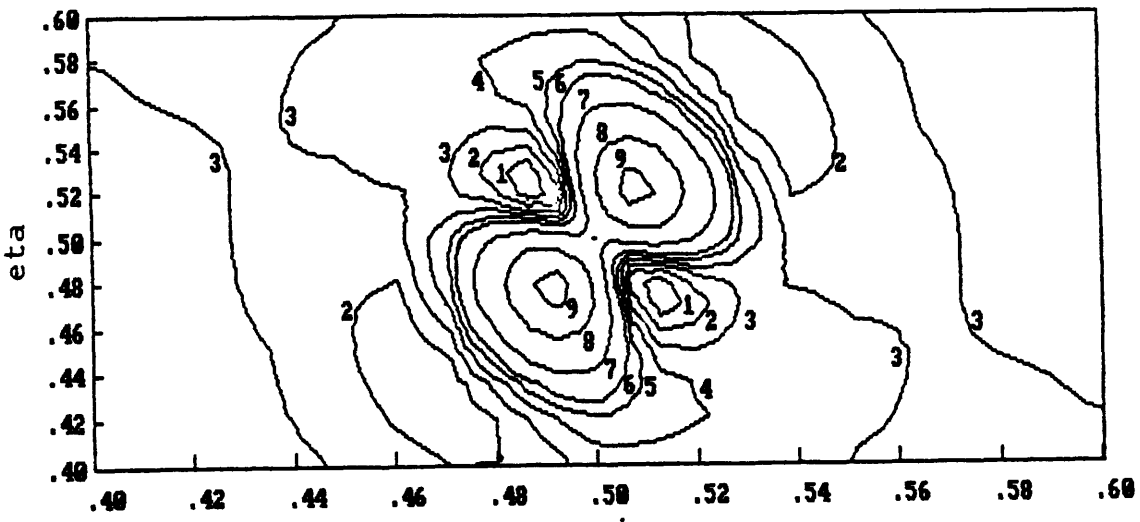


Figure 5.13 X-ray of a Damaged Specimen
(Magnification: 2x)

Figure 5.14 is a contour plot of predicted ε_1 strain in facesheet A at the instant of maximum indentation. Strains are shown in ply axes which, in this case, are oriented at 45° to the laminate axes. Contour 6 represents the level of 12000 μstrain, which is the value of ε_{ult}, the failure strain.



Key:	Contour:	Strain: (μstrain)
	1	-8000
	2	-4000
	3	0
	4	4000
	5	8000
	6	12000
	7	24000
	8	48000
	9	72000

Figure 5.14 Predicted Strain Contours

The amount of damage sustained by a specimen is quantified in the following manner: A line is drawn through the impact site parallel to the fibers (see Figure 5.15). This line follows the expected failure site: a tow boundary. The portion of this line which lies within the damage region is projected onto a line perpendicular to the loading direction. The damage width is the length of the projected line segment. In Figure 5.15, the analytical predictions of Figure 5.14 are superimposed on the observed damage in Figuer 5.13. The solid line represents the extent of damage seen in Figure 5.13. The dashed line represents the contours

corresponding to $\varepsilon_1=\varepsilon_{ult}$ and $\varepsilon_2=\varepsilon_{ult}$. In Figure 5.15, the experimental and analytical damage widths are labeled d_e and d_a respectively.

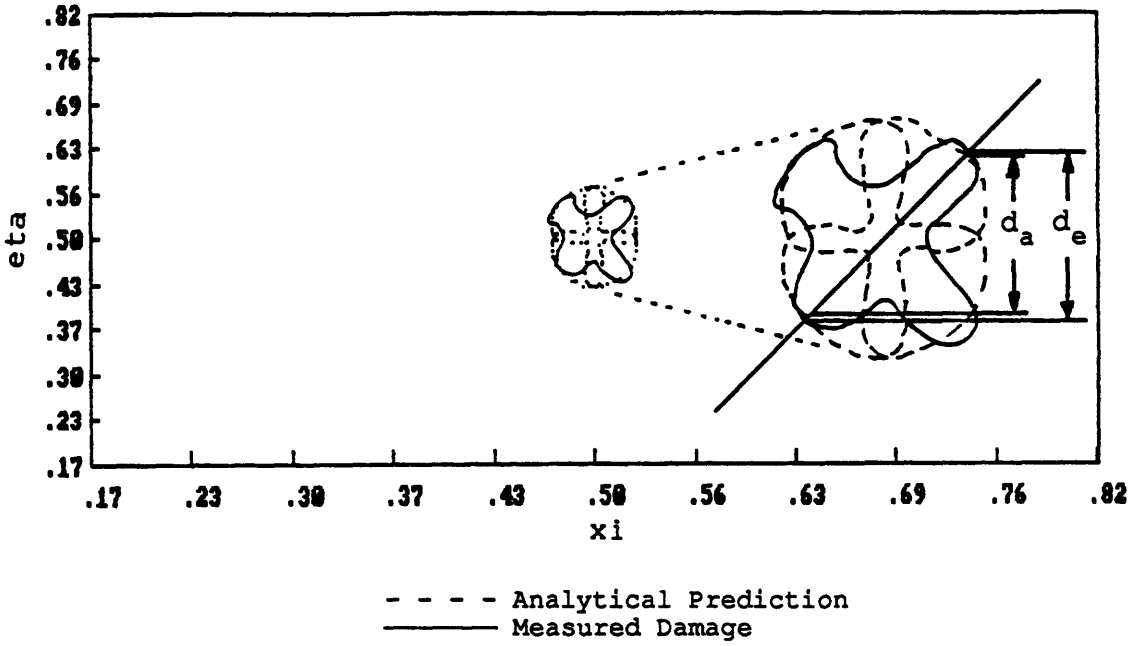


Figure 5.15 Damage Extent and Damage Prediction

Figure 5.16 is a graph of damage widths plotted against impact energy for various dynamic impacts. The symbols represent experimental values while the dashed lines represent the analytical predictions. The upper bound is the net-section result for a specimen with fibers at 45° to the principal load direction. The lower bound is the Mar-Lin prediction for a specimen with fibers parallel and perpendicular to the principal loading direction.

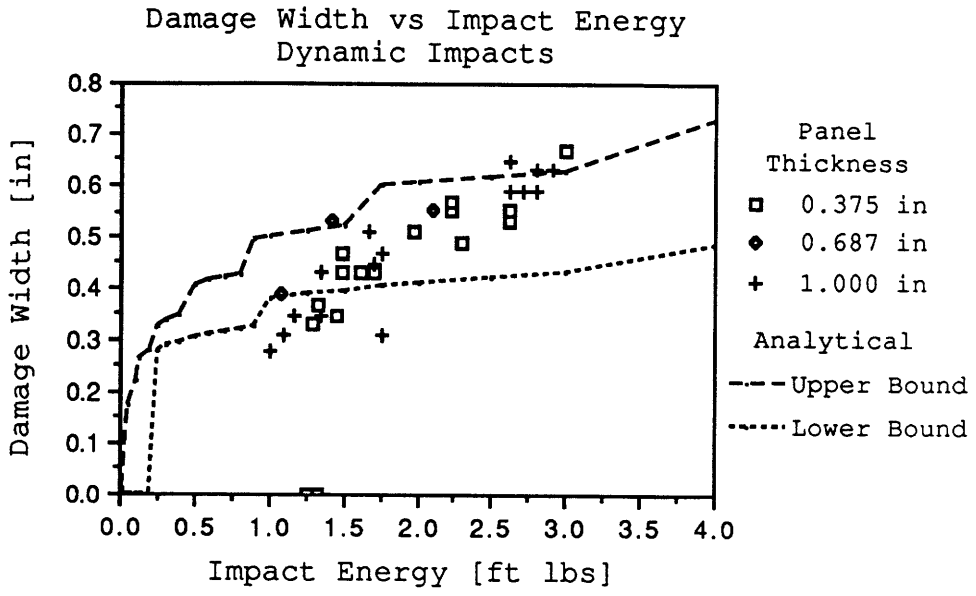


Figure 5.16 Damage Width vs Impact Energy

At lower energy levels, the analysis predicts more damage than actually occurs. In the energy range studied (1.4 to 3.0 ft lbs), the predictions match the experimental values well. Figure 5.17 is a graph of damage widths plotted against indentation depth for various static indentations. Again, the symbols represent experimental values while the dashed lines represent the analytical predictions.

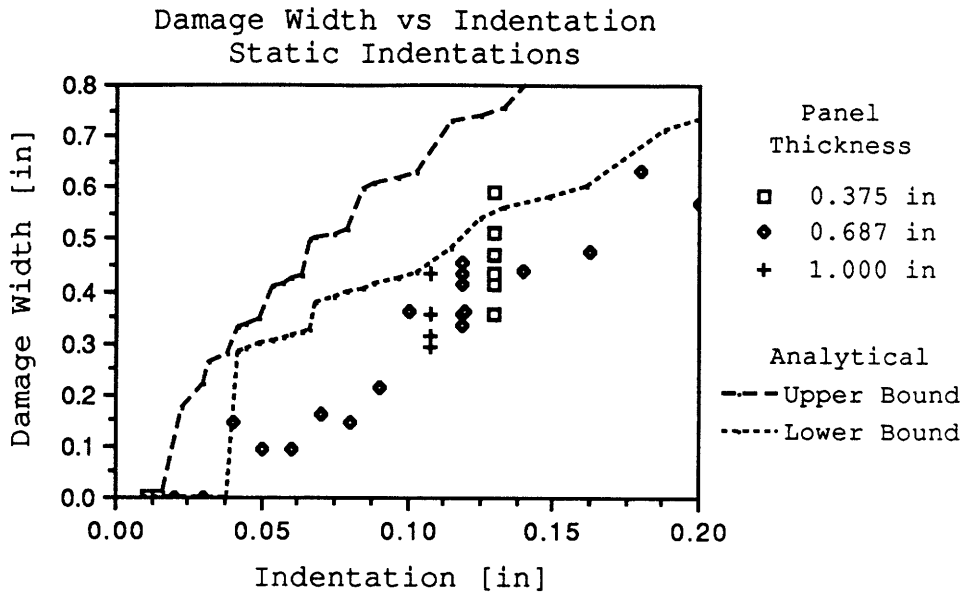


Figure 5.17 Damage Width vs Indentation

The analysis over predicts the amount of damage. This indicates a model which is too stiff. Any assumed modes solution is expected to be too stiff, and the use of only a single assumed mode[24] accentuates this problem.

5.10 Residual Strength - Undamaged Panels

Failure Modes

Three failure modes are observed for undamaged panels: facesheet fracture, core tearing, and core shear failure. Most specimens usually exhibited features from each of the three categories, but one mode always dominated.

[24]A single assumed mode is used for u, v, and w resulting in a 3 degree-of-freedom model. See Section 2.5 about the local model for more information.

Facesheet fracture usually occurred near one end of the specimen. This type of failure can be attributed to stress concentrations arising from improper load introduction. If this type of failure was observed, the test was considered invalid and the results were not used.

The second failure mode is core tearing. This is a tensile failure of the core material in the z direction. The core would separate in a region parallel to the facesheets near the film adhesive bond line. This would allow the facesheet to buckle outwards. The stresses necessary to tear the core would arise from the tendency of the facesheet to buckle locally. It is important to note that failure occurs within the core and that the film adhesive bond remains intact.

The third failure mode is core shear failure. In this mode, the core failed in shear in the x-z plane and the panel assumed as "S" shape as the now unsupported facesheets buckled. This type of failure, which was observed only in 3/8 inch thick specimens, would arise if the panel, rather than facesheets, started to buckle.

Table 5.18 shows the undamaged strengths for the different panel configurations.

Table 5.18 Undamaged Panel Strengths

Facesheet Layup	Core Thickness [in]	Experimental Failure Stress [ksi]	Analytical Failure Stress [ksi]
(0,90)	0.375	44.9 (3)	66.8
(0,90)	0.687	45.0 (6)	49.3
(0,90)	1.000	48.8 (2)	40.9
(±45)	0.375	27.8 (3)	47.4
(±45)	0.687	29.4 (3)	35.0
(±45)	1.000	30.3 (3)	29.0

Note: The numbers in parentheses indicate the number of specimens tested.

Several trends can be seen in the data in Table 5.18. First, the facesheet layup has a significant effect on panel strength. The (0,90) facesheets are stiffer than the (±45) facesheets in the loading direction and thus are better able to resist buckling. Second, the thicker panels are slightly stronger than the thinner panels. As the thicker panels are more resistant to overall Euler buckling, this behavior is to be expected. In the thinner panels, overall Euler buckling and local facesheet buckling can interact and cause an earlier failure. The residual strength model accounts only for local facesheet buckling and assumes that Euler buckling does not occur. Thus, it is fairly accurate for the thicker panels, but not as accurate for the thinner panels. The analytical prediction that the thinner panels are stronger than the thicker panels is due to an increase is the effective modulus of foundation as the core thickness decreases.

5.11 Residual Strength - Damaged Panels

In all except four cases, damaged panels failed by facesheet fracture through the impact site. The remaining four specimens failed due to facesheet fracture at one of the loading tabs. The four specimens which failed in this way were considered defective and the results were not used.

Figures 5.19 through 5.21 are photographs taken during a residual strength test. The dimple visible in the center of the specimen marks the impact sites. This particular specimen had been subjected to a dynamic impact of 1.20 ft lbs. The impact left a visible dimple in the specimen. During compression, the dimple grew in size and shape as the load increased. Initially, the dimple retained its shape but grew in area. Later, the dimple became elongated with the major axes perpendicular to the loading direction. This trend continued until final failure which resulted when the dimple grew across the entire panel and the facesheet fractured. This behavior is typical of all impacted and indented specimens.

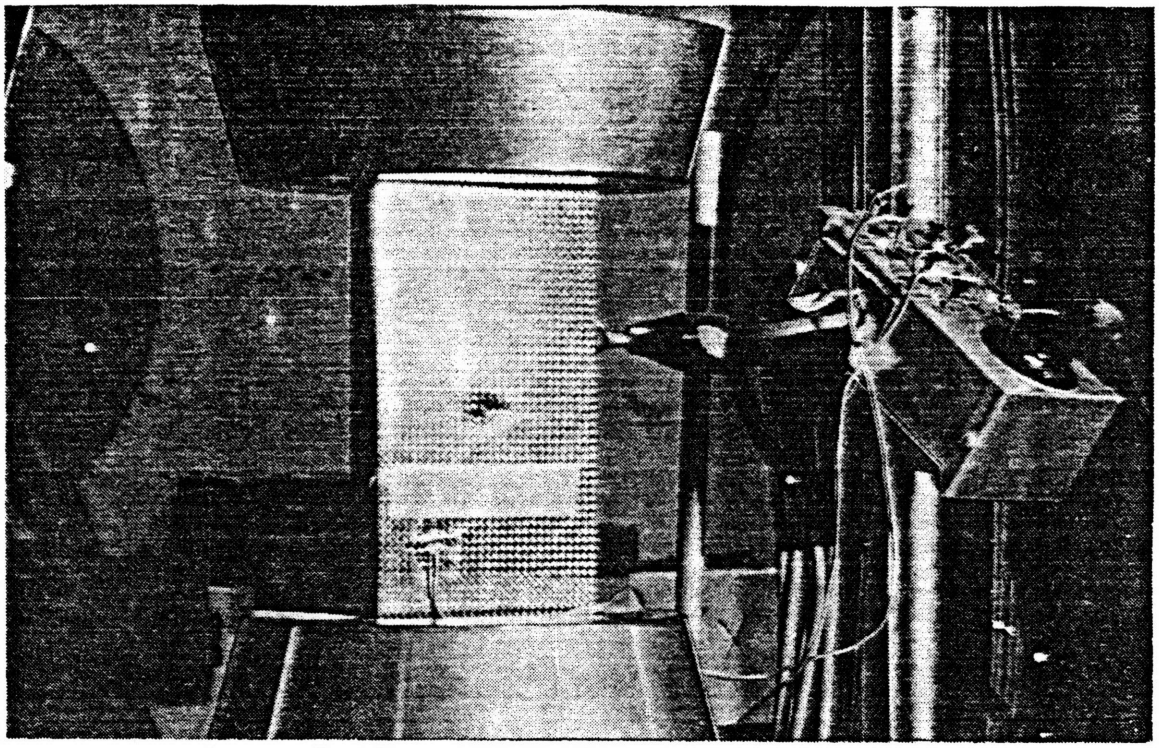


Figure 5.19 Compression Test - zero load

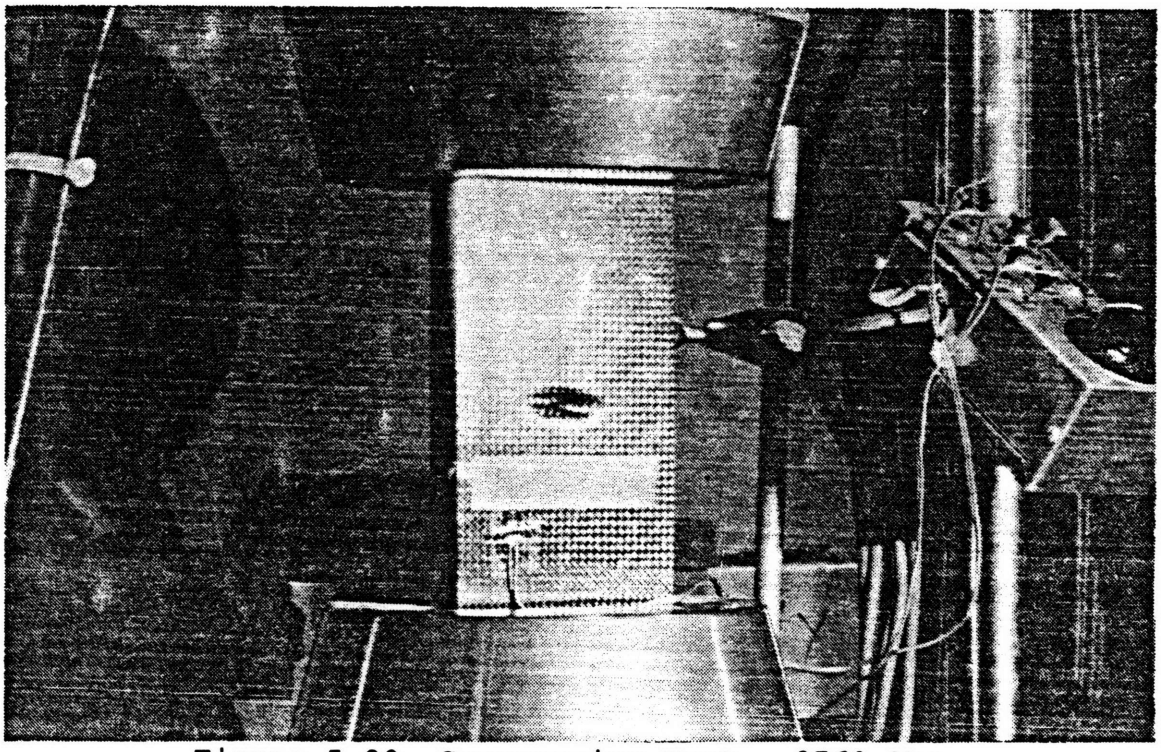


Figure 5.20 Compression Test - 2760 lbs

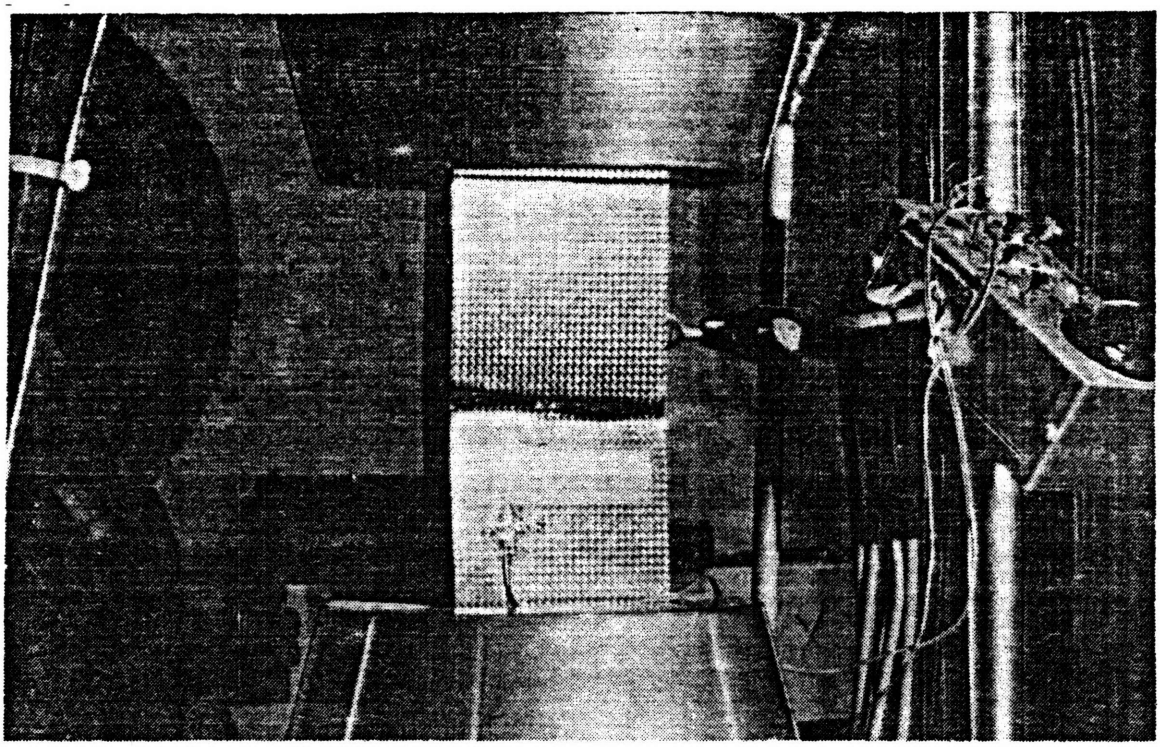


Figure 5.21 Compression Test - post failure

Post failure inspection reveals an interesting pattern of damage. The depth of core which remained attached to the facesheet varied along the length of the specimen. The variation was periodic with a wave length of approximately 1.2 inches. This suggests that the failure was initiated by facesheet buckling. Assuming this to be the case, the buckling mode number is calculated by:

$$\frac{\text{panel length}}{\text{half wave length}} = \frac{8.0 \text{ in}}{0.6 \text{ in}} = 13.33 = \sim 13 \qquad (80)$$

This agrees with the buckling mode number predicted by the analysis.

Figures 5.22 and 5.23 are graphs of failure stress versus damage width. Figure 5.22 represents specimens with (0,90) facesheets and Figure 5.23 represents specimens with (±45) facesheets. Experimental data as well as analytical predictions are included.

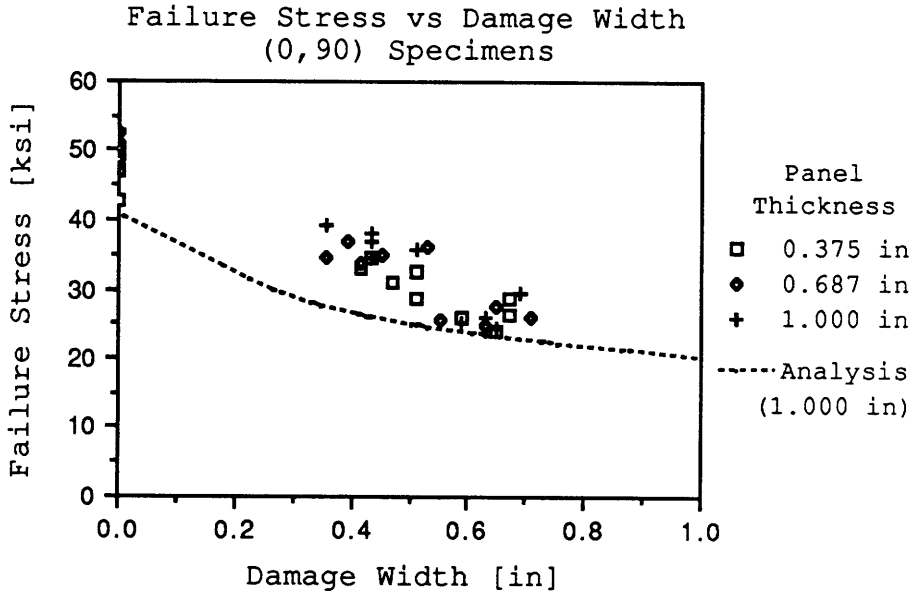


Figure 5.22 Failure Stress versus Damage Width (0,90) Specimens

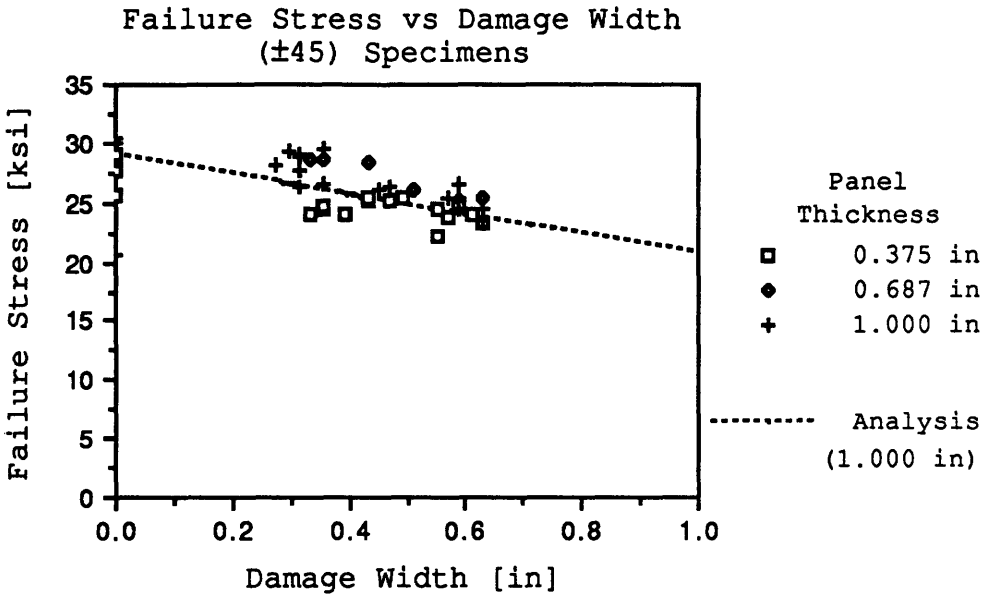


Figure 5.23 Failure Stress versus Damage Width (±45) Specimens

The analysis predicts the residual strength for the (±45) specimens quite well, but under predicts the residual strength of the (0,90) specimens. This is due to the over prediction of damage in the (0,90) specimens.

Recall that Figure 5.16 represents the relationships between impact energy and damage width. Combining the information in Figures 5.16, 5.22, and 5.23, it is possible to obtain the relationships between failure stress and impact energy. These relationships for (0,90) and (±45) specimens are shown in Figures 5.24 and 5.25, respectively.

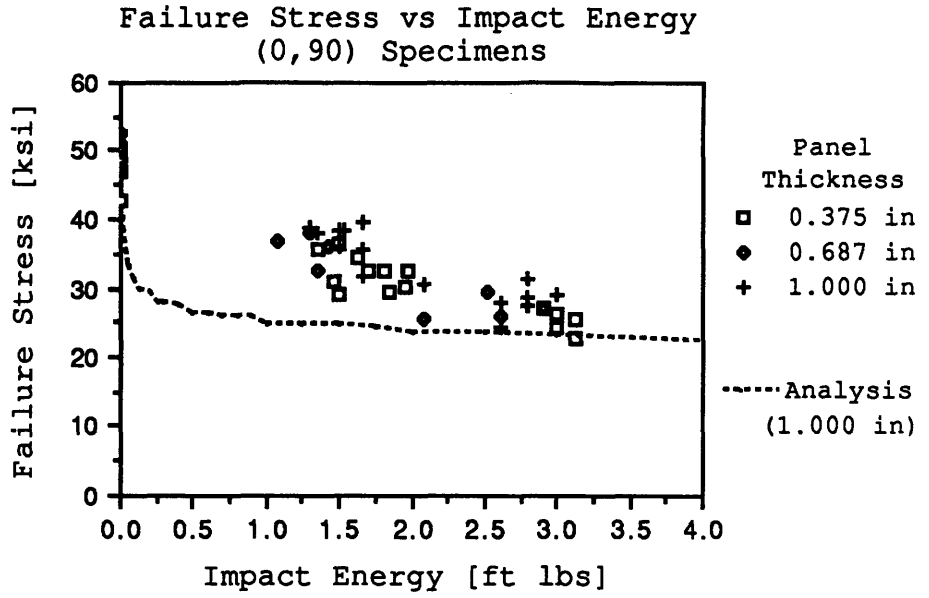


Figure 5.24 Failure Stress versus Impact Energy (0,90) Specimens

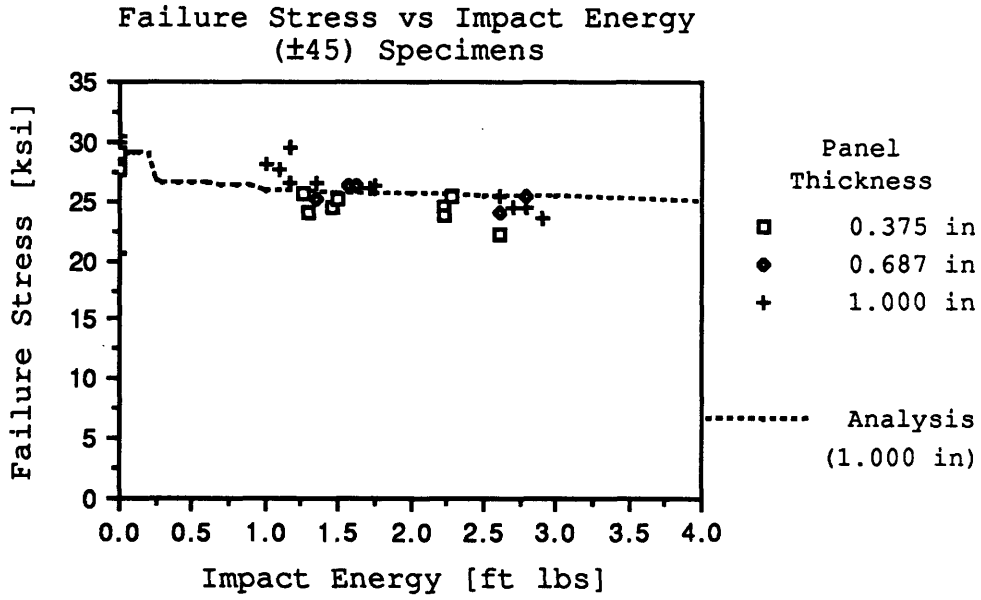


Figure 5.25 Failure Stress versus Impact Energy (±45) Specimens

Again, the analysis predicts the residual strength for the (±45) specimens quite well, but under predicts the residual strength of the (0,90) specimens.

6. Conclusions

The goals of this effort are two-fold: To describe the physics of low-velocity impacts to thin gauge composite honeycomb structures and to analytically predict the residual compressive strength of an impacted sandwich panel. The analysis is intended to be a preliminary design tool to allow engineers to obtain predictions of residual strength for panels subjected to various impacts. Both of these goals have been accomplished.

The following conclusions can be drawn concerning the behavior of two-ply specimens:

- Impact events below the level of 0.7 ft lbs of energy cause no appreciable damage.
- Damage occurs as core crushing (at 0.7 ft lbs), matrix cracks (at 1.4 ft lbs), fiber breakage (1.5 ft lbs), and delamination (1.5 ft lbs).
- Static indentations cause the same types of damage as dynamic impacts. Thus, dynamic impacts under these conditions can be considered quasi-static. This is due to the relatively high impactor mass and the stiffnesses of the panel and Hertzian spring.
- Damage is limited to a circular area with a diameter of about 150% of the impactor diameter. Higher energy impacts to not cause a larger damage area, but rather, more extensive damage in the same area.
- Panel failure is caused by local buckling of one (or both) facesheets. This phemonena is also known as

"facesheet wrinkling." The film adhesive and core contribute to the bending stiffness of the facesheet and this must be accounted for in the analysis.

- The residual strength of the panels depends upon the facesheet layup. Panels with (±45) facesheets are notch insensitive and fail at a nearly constant value of net-section stress. Panels with (0,90) facesheets are notch sensitive and the failure stress is lower than that predicted by constant net-section stress.

The analysis procedure has been implemented as a series of FORTRAN computer programs. These programs together comprise a design tool for use in analysis of low-velocity impacts. The following conclusions can be drawn concerning the analytical procedure:

- The global model can predict the overall response of the panel to impact. It has sufficient detail to model the important physics of the problem and can be used to predict the contact force between the panel and the impactor.

- The global model exploits the fact that most sandwich structures are manufactured with relatively thin facesheets. As long as these assumptions are met, the model works well. However, these assumptions may break down for thicker facesheets.

- The local model has sufficient detail to account for the physics of the problem studied and can predict strains caused by the impact event.

- The local model does not account for impactor geometry.

- The panel buckling model agrees with experimental data for the panels tested. It is based on a simple beam model and incorporates the effects of film adhesive and core on failure stress.

- The damage prediction is a simple point-stress criteria utilizing ε_1 in ply axes. The analysis over predicts damage at extremely low energy levels (<0.5 ft lbs), but otherwise agrees with experimental data for the specimens tested.

- The residual strength model uses net-section stress and the Mar-Lin relation to predict residual strength.

The analytical model was developed to be a working tool to aid designers by giving preliminary information about the amount of damage caused by a low-velocity foreign-body impact and the residual strength of the damaged panel. In this regard, the analysis is successful. The analysis can predict the amount of damage as well as the residual strength. In addition, the computational effort required is low and thus the analysis inexpensive.

The work here suggests additional area of investigation:

- The local model could be modified to account for impactor geometry. Currently, the local model requires that the panel displacement match the impactor only at the center point. The model could be modified to require that the displacements match at several or all points.

- The current panel buckling analysis is based on a simple beam model. This analysis could be expanded into a full-plate buckling problem.

- The present damage prediction method is a simple point-stress criteria. This could be expanded to a more involved criteria considering several stresses. This could resolve the problem of inaccurate damage predictions at low energy levels

- The residual strength model currently uses net-section stress and the Mar-Lin relation. This model could be expanded into a analytical model of a damaged plate. This could improve the accuracy of the residual strength predictions.

- Different specimen configurations could be tested to explore range of validity of the analysis.

In conclusion, the physics of the impact problem have been studied and an analytical model has been developed. Good agreement has been found between the experimental data and the analytical model. The results of this model may not be as accurate as the results of other models, but it was designed as a fast-preliminary analysis tool and in this regard, it is successful.

References

1 K.J. Bathe. Numerical Methods in Finite Element Analysis. Prentice-Hall, Inc. New Jersey, 1982.

2 R.L. Bisplinghoff and H. Ashley. Principles of Aeroelasticity. Dover Publications, Inc. New York, 1962.

3 D.S. Cairns. Impact and Post-Impact Response of Graphite/Epoxy and Kevlar/Epoxy Structures. Ph.D. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology. August 1987.

4 J. Dugundji. *Simple Expressions for Higher Vibration Modes of Uniform Euler Beams.* Technical Note, AIAA Journal, Volume 26, No. 8. August 1988.

5 C.E. Federson. *Evaluation and Prediction of the Residual Strength of Center Cracked Tension Panels.* Damage Tolerance in Aircraft Structures. ASTM Special Technical Publication 486. 1970.

6 M.J. Graves and J. Koontz. *Initiation and Extent of Impact Damage in Graphite/Epoxy and Graphite/PEEK Composites.* AIAA 88-2327, Proceedings of the 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA. 1988.

7 R.M. Jones. Mechanics of Composite Materials. Scripta Book Company. Washington, DC, 1975.

8 P.A. Lagace. *Notch Sensitivity of Graphite/Epoxy Fabric Laminates.* Composites Science and Technolgy, Volume 26. 1986.

9 K.Y. Lin. Fracture of Filamentary Composite Materials. Ph.D. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology. 1977.

10 W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. Numerical Recipes - The Art of Scientific Computing. Cambridge University Press. Cambridge, 1986.

11 R.M. Rivello. Theory and Analysis of Flight Structures. McGraw-Hill, Inc. New York, 1969.

12 S.P. Timoshenko and J.N. Goodier. Theory of Elastic Stability. Third Edition, McGraw-Hill, Inc. Singapore, 1982.

13 S.P. Timoshenko and S. Woinowsky-Krieger. Theory of Plates and Shells. McGraw-Hill Book Company, Inc. New York, 1959.

Appendix A: FORTRAN Source Code

FORTRAN source code for the analysis programs is included on the following pages. These programs were compiled with the Microsoft FORTRAN 4.1 Optimizing Compiler. The Microsoft utility "MAKE" was used to organize compilation of the programs. The "MAKE" data files are included and show the options used when compiling the programs.

Program:	Required Source Code Files:
PROBLEM	PROBLEM.FOR, SPAD.INC
HZ	HZ.FOR, SPAD.INC, LISANR.FOR
INTGL	INTGL.FOR, QROMB.FOR, SPAD.INC
SPAD	SPAD.FOR, SPADRT.FOR, SPADBF.FOR, SPADLP.FOR, SPAD.INC
LISA	LISA.FOR, LISANR.FOR, SPAD.INC
DIFA	DIFA.FOR, LISANR.FOR, SPAD.INC

```
#This is a MAKE file to update problem.exe as necessary

problem.exe: problem.for spad.inc
    fl /FP problem.for



#This is a MAKE file to update hz.exe as necessary

hz.obj: hz.for spad.inc
    fl /FPa /Gt /Od /c hz.for

lisanr.obj: lisanr.for spad.inc
    fl /FPa /Gt /Od /c lisanr.for

hz.exe: hz.obj lisanr.obj
    fl /FPa /Gt /Od hz lisanr



#This is a MAKE file to update intgl.exe as necessary

intgl.obj: intgl.for spad.inc
    fl /FPa /c /Zi intgl.for

qromb.obj: qromb.for
    fl /FPa /c /Zi qromb.for

intgl.exe: intgl.obj qromb.obj
    fl /FPa /Zi intgl qromb
```

```
#This is a MAKE file to update spad.exe as necessary
.for.obj:
    fl /c /FPa /Gt $*.for

spad.obj: spad.for spad.inc

spadrt.obj: spadrt.for spad.inc

spadbf.obj: spadbf.for spad.inc

spadlp.obj: spadlp.for

spad.exe: spad.obj spadrt.obj spadbf.obj spadlp.obj
    fl /FPa /Gt spad spadrt spadbf spadlp
```

```
#This is a MAKE file to update lisa.exe as necessary

lisa.obj: lisa.for spad.inc
    fl /Gt /Od /c lisa.for

lisanr.obj: lisanr.for spad.inc
    fl /Gt /Od /c lisanr.for

lisa.exe: lisa.obj lisanr.obj
    fl /Gt /Od lisa lisanr
```

```
#This is a MAKE file to update difa.exe as necessary

difa.obj: difa.for spad.inc
    fl /Gt /Od /c difa.for

lisanr.obj: lisanr.for spad.inc
    fl /Gt /Od /c lisanr.for

difa.exe: difa.obj lisanr.obj
    fl /Gt /Od difa lisanr
```

```
C  File:  PROBLEM.FOR
       program problem
C  Program PROBLEM generates input files for use with
C  analysis program.  These files are identified by having
C  no extension.

       implicit real*8 (a-h,o-z)
$include:'spad.inc'
       integer bcx,bcy
       dimension diam(4)

       write (*,10)
10     format (' Sandwich Panel Impact Analysis Program'/
     &  ' Input File Generation Program'//)

       write (*,20)
20     format (' -- Panel Geometry --'/
     &  ' Enter panel length [in]: '\)
       read (*,*) a
       write (*,30)
30     format (' Enter panel width [in]: '\)
       read (*,*) b
       write (*,40)
40     format (' The choices for boundary conditions are:'/
     &  '    1.  Simply Supported - Simply Supported'/
     &  '    2.  Clamped - Free'/
     &  '    3.  Clamped - Clamped'/
     &  '    4.  Free - Free'/
     &  '    5.  Simply Supported - Clamped'/
     &  '    6.  Simply Supported - Free'//
     &  ' Enter code for the end',
     &  ' boundary conditions [1-6]: '\)
       read (*,*) bcx
       write (*,50)
50     format (' Enter code for the side',
     &  ' boundary conditions [1-6]: '\)
       read (*,*) bcy

       write (*,60)
60     format (/' -- Facesheet A --'/
     &  ' Enter A11 [lbs/in]: '\)
       read (*,*) aa(1)
       write (*,70)
70     format (' Enter A22 [lbs/in]: '\)
       read (*,*) aa(2)
       write (*,80)
80     format (' Enter A12 [lbs/in]: '\)
       read (*,*) aa(3)
       aa(4)=0.0
       aa(5)=0.0
       write (*,110)
110    format (' Enter A66 [lbs/in]: '\)
       read (*,*) aa(6)
       write (*,120)
```

```
120     format (' Enter facesheet thickness [in]: '\)
        read (*,*) ta
        write (*,130)
130     format (' Enter facesheet density [pci]: '\)
        read (*,*) rhoa
C  Convert from lbs/in^3 to slugs/in^3
        rhoa=rhoa/32.17404855

        write (*,140)
140     format (/' -- Facesheet B --'/
     &  ' Enter A11 [lbs/in]: '\)
        read (*,*) ab(1)
        write (*,70)
        read (*,*) ab(2)
        write (*,80)
        read (*,*) ab(3)
        ab(4)=0.0
        ab(5)=0.0
        write (*,110)
        read (*,*) ab(6)
        write (*,120)
        read (*,*) tb
        write (*,130)
        read (*,*) rhob
C  Convert from lbs/in^3 to slugs/in^3
        rhob=rhob/32.17404855

        write (*,150)
150     format(/' -- Core --'/)
        write (*,160) 'Ezz'
160     format (' Enter ',a3,' [psi]: '\)
        read (*,*) Ezz
        write (*,160) 'Gxz'
        read (*,*) Gxz
        write (*,160) 'Gyz'
        read (*,*) Gyz
        write (*,170)
170     format (' Enter core thickness [in]: '\)
        read (*,*) h
        write (*,180)
180     format (' Enter core density [pcf]: '\)
        read (*,*) rhoc
C  Convert from lbs/ft^3 to slugs/in^3
        rhoc=rhoc/32.17404855/12/12/12

        write (*,190)
190     format (/' -- Film Adhesive --'/
     &  ' Enter adhesive density [psf]: '\)
        read (*,*) rhof
        tf=.01
C  Convert from lbs/ft^2 to slugs/in^3
        rhof=rhof/32.17404855/12/12/tf

        write (*,200)
```

```
200     format (/' -- Impact Event --'/
     &  ' Enter impactor weight [lbs]: '\)
        read (*,*) mi
C  Convert from lbs to slugs and then to lbs sec^2/IN
        mi=mi/32.17404855/12.0
        write (*,210)
210     format (' Enter impact energy [ft lbs]: '\)
        read (*,*) e
C  What we need is initial velocity
        v=sqrt(2.0*e/(mi*12.0))
C  That's in ft/s so convert to in/s
        v=v*12.0
        write (*,215)
215     format (' Enter impactor diameter [in]: '\)
        read (*,*) dia
C  Radius is half of diameter
        ri=dia/2

        write (*,220)
220     format (/' -- Analysis Parameters --'/)
        hk=0.0d0
        write (*,230) 'length'
230     format (' Enter number of modes along the ',
     &          a,' [1-6]: '\)
        read (*,*) nx
        write (*,230) 'width'
        read (*,*) ny
        write (*,240)
240     format (' Enter time step [sec]: '\)
        read (*,*) dt

245     write (*,250)
250     format (/' Enter Filename: '\)
        read (*,260) fname
260     format (a40)

        open (unit=2,file=fname)
        write (2,270) fname
270     format (1x,a40)
        write (2,*) a,b,bcx,bcy
        write (2,*) (aa(i),i=1,6)
        write (2,*) ta,rhoa
        write (2,*) (ab(i),i=1,6)
        write (2,*) tb,rhob
        write (2,*) Ezz,Gxz,Gyz,h,rhoc
        write (2,*) rhof,tf
        write (2,*) mi,v,hk,ri
        write (2,*) nx,ny
        write (2,*) dt
        close (unit=2)

        write (*,280) fname
280     format (/' File: ',a,' has been written.')
        end
```

```
C  File:  SPAD.INC
      parameter (maxnx=6,maxny=6,iodd=1,
     &   maxdim=maxnx*maxny*6,maxvec=maxdim*(maxdim+1)/2)
      real*8 mi
      integer rank,q1beg,q1end,q4beg,q4end
      character*40 fname
      common /plate/ aa(6),ab(6),ta,tb,rhoa,rhob,Ezz,
     &   Gxz,Gyz,h,a,b,area,rhoc,rhof,tf,zac,zbc,
     &   za,zb,nm(2),nv,rank,
     &   psi(0:2,0:2,maxnx,maxnx),phi(0:2,0:2,maxny,maxny),
     &   bcbeta(maxnx,2),bcthet(2),bca(2),bcb(maxnx,2)
      common /impact/ mi,v,dt,hk,q1beg,q1end,q4beg,q4end,
     &   ri,di,al,ah,bl,bh,fname,imodes
```

```
C  File:  HZ.FOR
        program hz
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer bcx,bcy
        real*8 intgl,lsh
        logical success
        dimension q(maxdim),modes(3,maxdim),talpha(9),teng(9)
        common /local/ alpha,intgl(0:2,0:2,2,0:2,0:2,2),
     &         supint(5),bm,cm
        common /local2/ bintgl(6,2,6,2,0:8),bsup(5,0:8)
        external streng,strain
        data talpha /.005, .01, .02, .03, .04,
     &          .05, .06, .07, .08/

        write (*,10)
10      format (/'  -- Hertzian Spring Constant Analysis --')

        write (*,*) 'Reading input files . . .'
        call input(bcx,bcy)
        call getint(bintgl,bsup)

        write (*,*) 'Calculating indentations . . .'

        x1=5.0
        x2=4.0
        ftol=1.0d-8

C  Solve for the local indentation

        write (*,*)
        do 40 it=1,9
          alpha=talpha(it)
          call mnbrak(x1,x2,x3,f1,f2,f3,streng)
          write (*,15) it
15        format ('+',i2,'/9 ')
          fret=brent(x1,x2,x3,streng,ftol,xmin)
          teng(it)=fret
          write (*,20) it
20        format ('+',i2,'/9*')
40      continue
        write (*,*)

C  Fit the data to the Hertzian model:  f=k[alpha]^3/2
C                            or    Wext=2/5 k [alpha]^5/2
        t1=0.0
        t2=0.0
        do 50 i=1,9
          t1=t1+teng(i)*talpha(i)**2.50d0
          t2=t2+talpha(i)**5.0d0
50      continue
        hk=2.5d0*t1/t2

C  And write the results
```

```
        call output(bcx,bcy)
        write (*,60) fname
60      format (1x,a8,' updated')
        end

C-------------------------------------------------------------
C
C  Function STRENG returns the strain energy given a value
C  of the shape parameter x.

        real*8 function streng(x)
        implicit real*8 (a-h,o-z)
        real*8 intgl
        integer*1 lookup
        dimension sp(0:8)
        dimension lookup(0:2,0:2)
        dimension p(3,2),y(3),q(2)
        common /local/ alpha,intgl(0:2,0:2,2,0:2,0:2,2),
     &         supint(5),bm,cm
        common /local2/ bintgl(6,2,6,2,0:8),bsup(5,0:8)
        external func
        data lookup /1,2,4,3,6,0,5,0,0/

C  Calculate powers of the shape parameter

        sp(0)=1.0d0/(x*x)
        do 100 i=1,8
          sp(i)=sp(i-1)*x
100     continue

C  Generate the new intgl values for this value of the
C  shape parameter

        do 120 nx=0,2
        do 120 ny=0,2
        do 120 ip=1,2
        do 120 mx=0,2
        do 120 my=0,2
        do 120 jp=1,2
          if (nx+ny.le.2.and.mx+my.le.2) then
            t=0.0
            do 110 k=0,8
              t=t+bintgl(lookup(nx,ny),ip,
     &           lookup(mx,my),jp,k)*sp(k)
110         continue
            intgl(nx,ny,ip,mx,my,jp)=t
          endif
120     continue

C  Don't forget supint values as well

        do 140 i=1,5
          t=0.0
```

```
          do 130 k=0,8
            t=t+bsup(i,k)*sp(k)
130       continue
          supint(i)=t
140     continue

C  Minimize over the u and v variables (given sp).

        t=1.0d-4
        p(1,1)=t
        p(1,2)=0.0d0
        p(2,1)=0.0d0
        p(2,2)=t
        p(3,1)=t
        p(3,2)=t
        do 150 i=1,3
          q(1)=p(i,1)
          q(2)=p(i,2)
          y(i)=func(q)
150     continue
        ftol=1e-6
        call amoeba(p,y,3,2,2,ftol,func,iter)

        ilo=1
        do 160 i=2,3
          if (y(i).lt.y(ilo)) ilo=i
160     continue
        streng=y(ilo)
        bm=p(ilo,1)
        cm=p(ilo,2)
        return
        end

C-------------------------------------------------------------
C
C  Function FUNC is the total strain energy
C  This routine returns U as a function of bi=ci and
C  assumes that the integral tables have been updated
C  to the curren value of the shape parameter sp.

        real*8 function func(x)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        real*8 intgl
        dimension x(2)
        common /local/ alpha,intgl(0:2,0:2,2,0:2,0:2,2),
     &         supint(5),bm,cm

        z=za
        z2=z*z
        z3=z2*z
        z4=z3*z

        ai=alpha
```

```
      bi=x(1)
      ci=x(2)

      a2=ai*ai
      b2=bi*bi
      c2=ci*ci
      t=0.0d0

C  U1:  terms with A11
        t=t+aa(1)/2*(z2*bi*bi*intgl(2,0,1,2,0,1)
     &  + z3*bi*b2*intgl(2,0,1,2,0,2)
     &  + z3*bi*c2*intgl(2,0,1,1,1,2)
     &  + z*bi*a2*intgl(2,0,1,1,0,2)
     &  + z4/4*b2*b2*intgl(2,0,2,2,0,2)
     &  + z4/2*b2*c2*intgl(2,0,2,1,1,2)
     &  + z2/2*b2*a2*intgl(2,0,2,1,0,2)
     &  + z4/4*c2*c2*intgl(1,1,2,1,1,2)
     &  + z2/2*c2*a2*intgl(1,1,2,1,0,2)
     &  + a2*a2/4*intgl(1,0,2,1,0,2))

C  U2:  terms with A12
        t=t+aa(3)*(z2*bi*ci*intgl(2,0,1,0,2,1)
     &  + z3/2*bi*b2*intgl(2,0,1,1,1,2)
     &  + z3/2*bi*c2*intgl(2,0,1,0,2,2)
     &  + z/2*bi*a2*intgl(2,0,1,0,1,2)
     &  + z3/2*b2*ci*intgl(2,0,2,0,2,1)
     &  + z4/4*b2*b2*intgl(2,0,2,1,1,2)
     &  + z4/4*b2*c2*intgl(2,0,2,0,2,2)
     &  + z2/4*b2*a2*intgl(2,0,2,0,1,2)
     &  + z3/2*c2*ci*intgl(1,1,2,0,2,1)
     &  + z4/4*c2*b2*intgl(1,1,2,1,1,2)
     &  + z4/4*c2*c2*intgl(1,1,2,0,2,2)
     &  + z2/4*c2*a2*intgl(1,1,2,0,1,2)
     &  + z/2*a2*ci*intgl(1,0,2,0,2,1)
     &  + z2/4*a2*b2*intgl(1,0,2,1,1,2)
     &  + z2/4*a2*c2*intgl(1,0,2,0,2,2)
     &  + a2*a2/4*intgl(1,0,2,0,1,2))

C  U3:  terms with A22
        t=t+aa(2)/2*(z2*ci*ci*intgl(0,2,1,0,2,1)
     &  + z3*ci*b2*intgl(0,2,1,1,1,2)
     &  + z3*ci*c2*intgl(0,2,1,0,2,2)
     &  + z*ci*a2*intgl(0,2,1,0,1,2)
     &  + z4/4*b2*b2*intgl(1,1,2,1,1,2)
     &  + z4/2*b2*c2*intgl(1,1,2,0,2,2)
     &  + z2/2*b2*a2*intgl(1,1,2,0,1,2)
     &  + z4/4*c2*c2*intgl(0,2,2,0,2,2)
     &  + z2/2*c2*a2*intgl(0,2,2,0,1,2)
     &  + a2*a2/4*intgl(0,1,2,0,1,2))

C  U4 and U5 are zero due to orthotropic facesheets

C  U6:  terms with A66
```

```
          t1=z2*bi*(bi+ci)*intgl(1,1,1,1,1,1)
     &   + 2*z2*bi*(b2+c2)*supint(1)
     &   + 2*z*bi*a2*supint(2)
     &   + z2*ci*ci*intgl(1,1,1,1,1,1)
     &   + 2*z3*ci*(b2+c2)*supint(1)
     &   + 2*z*ci*a2*supint(2)
          t1=t1+z4*b2*b2*supint(3)
     &   + 2*z4*b2*c2*supint(3)
     &   + 2*z2*b2*a2*supint(4)
     &   + z4*b2*b2*supint(3)
     &   + 2*z2*c2*a2*supint(4)
     &   + a2*a2*supint(5)
          t=t+aa(6)/2*t1

C  U7:  terms with Ezz
          t=t+Ezz/2/h*ai*ai*intgl(0,0,1,0,0,1)

C  U8:  terms with Gyz
          t=t+Gyz*h/2*(ai*ai*intgl(0,1,1,0,1,1)/3
     &   + ai*bi*intgl(0,1,1,1,0,1)*5/12
     &   + bi*bi*intgl(1,0,1,1,0,1)/2)

C  U9:  terms with Gxz
          t=t+Gyz*h/2*(ai*ai*intgl(1,0,1,1,0,1)/3
     &   + ai*ci*intgl(1,0,1,0,1,1)*5/12
     &   + ci*ci*intgl(0,1,1,0,1,1)/2)

          func=t
          return
          end

C-------------------------------------------------------------
C
C  Subroutine FILEXT appends the extension ext to name.
C
          subroutine filext(name,ext)
          character*40 name
          character*4 ext
          j=0
          do 10 i=1,36
            if (name(i:i).eq.'.'
     &          .or.name(i:i).eq.' '.and.j.eq.0) j=i
10        continue
          if (j.eq.0) then
            write (*,*) 'File name too long.'
            stop
          endif

          name(j:j+3)=ext(1:4)
          name(j+4:j+4)=' '
          return
          end

C-------------------------------------------------------------
```

```
C
C  Subroutine INPUT reads data from a prepared file for use
C  by the analysis programs SPAD and LISA.
C  These data files are created by PROBLEM.FOR and have
C  no extension.

      subroutine input(bcx,bcy)
      implicit real*8 (a-h,o-z)
      integer bcx,bcy
$include:'spad.inc'
      character*40 line
      pi=3.14159265358979323846

      open(unit=2,file=' ',err=110)
      read (2,10) line
10    format (a40)
      j=1
20    j=j+1
      if (line(j:j).eq.' '.or.line(j:j).eq.09) goto 20
      k=j
30    k=k+1
      if (line(k:k).ne.' ') goto 30
      line(k:k)='.'
      fname(1:k-j+1)=line(j:k)
      do 40 i=k-j+2,40
        fname(i:i)=' '
40    continue
      read (2,*,err=100) a,b,bcx,bcy
      read (2,*,err=100) (aa(i),i=1,6)
      read (2,*,err=100) ta,rhoa
      read (2,*,err=100) (ab(i),i=1,6)
      read (2,*,err=100) tb,rhob
      read (2,*,err=100) Ezz,Gxz,Gyz,h,rhoc
      read (2,*,err=100) rhof,tf
      read (2,*,err=100) mi,v,hk,ri
      read (2,*,err=100) nm(1),nm(2)
      read (2,*,err=100) dt
      close (unit=2)
      zac=h/2
      zbc=-zac
      za=zac+ta/2
      zb=zbc-tb/2
      area=a*b
      di=sqrt(pi*ri*ri)
      al=0.5-di/2/a
      ah=0.5+di/2/a
      bl=0.5-di/2/b
      bh=0.5+di/2/b
      return

100   write (*,105) fname
105   format (' Error reading data from file: ',a40)
      goto 120
110   write (*,115) fname
```

```
115     format (' Error opening file: ',a40)
120     stop
        end

C-------------------------------------------------------------
C
C  Subroutine OUTPUT updates the Hertzian Spring Constant
C  in the description file.

        subroutine output(bcx,bcy)
        implicit real*8 (a-h,o-z)
        integer bcx,bcy
$include:'spad.inc'

        call filext(fname,'    ')
        open(unit=3,file=fname,status='old')
        write (3,10) fname
10      format (' ',a40)
        write (3,*) a,b,bcx,bcy
        write (3,*) (aa(i),i=1,6)
        write (3,*) ta,rhoa
        write (3,*) (ab(i),i=1,6)
        write (3,*) tb,rhob
        write (3,*) Ezz,Gxz,Gyz,h,rhoc
        write (3,*) rhof,tf
        write (3,*) mi,v,hk,ri
        write (3,*) nm(1),nm(2)
        write (3,*) dt
        close (unit=3)
        return
        end

C-------------------------------------------------------------
C
C  Subroutine GETQ reads in the global plate displacements
C  and maximum approach generated by the SPAD program.

        subroutine getq(q,alpha,modes)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        dimension q(maxdim),modes(3,maxdim)
        call filext(fname,'.tq ')
        open (unit=2,file=fname,status='old')
        read (2,*) alpha,rank
        read (2,*) nm(1),nm(2)
        do 20 i=1,rank
          read (2,*) (modes(j,i),j=1,3),q(i)
20      continue
        do 30 i=1,2
          read (2,*) (bcbeta(j,i),j=1,nm(i))
          read (2,*) bcthet(i),bca(i)
          read (2,*) (bcb(j,i),j=1,nm(i))
30      continue
        close (unit=2)
```

```
        return
        end

C-------------------------------------------------------------
C
C  Subroutine GETINT reads the table of integrals from the
C  INTGL program.

        subroutine getint(intgl,supint)
        implicit real*8 (a-h,o-z)
        real*8 intgl
        integer pi,pj
        dimension intgl(6,2,6,2,0:8)
        dimension supint(5,0:8)

        open (unit=2,file='intgl.tbl',status='old')
        do 40 i=1,6
        do 40 pi=1,2
        do 40 j=1,i
        do 40 pj=1,2
          read (2,*) (intgl(i,pi,j,pj,n),n=0,8)
          do 30 k=0,8
            intgl(j,pj,i,pi,k)=intgl(i,pi,j,pj,k)
30        continue
40      continue

        do 50 i=1,5
          read (2,*) (supint(i,n),n=0,8)
50      continue

        close (unit=2)
        return
        end

C-------------------------------------------------------------
C
C  Function SHAPE returns the value of:
C       the pth derivative of beam function n
C       in the i direction at point xi (normalized).

        real*8 function shape(i,n,p,x)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer i,n,p

        r2=sqrt(2.0)
        beta=bcbeta(n,i)
        if (beta.eq.0) then
          if (p.eq.0) then
            shape=1.0
          else
            shape=0.0
          endif
        else
```

```
            theta=beta*x+bcthet(i)
            e1=bca(i)*exp(-beta*x)
            e2=bcb(n,i)*exp(-beta*(1.0-x))
            if (p.eq.0) then
              shape=r2*dsin(theta)+e1+e2
            elseif (p.eq.1) then
              shape=beta*(r2*dcos(theta)-e1+e2)
            else
              shape=beta*beta*(-r2*dsin(theta)+e1+e2)
            endif
          endif
          return
          end

C-------------------------------------------------------------
C
C  Function LSH returns the local impact mode shape function.
C    The value returned is the ixth x derivative times the
C    iyth y derivative of the shape function (with shape
C    parameter sp) at the normalized point (xi,eta).
C    These coordinates are normalized [0,1] over the plate.

          real*8 function lsh(ix,iy,xi,eta,sp)
          implicit real*8 (a-h,o-z)
$include:'spad.inc'
          integer*1 ak(7,2)
          dimension t(2)

          x=a*(xi-0.5d0)
          y=b*(eta-0.5d0)
          r=sqrt(x*x+y*y)
          z=sp*r
          call defshp(ix,iy,ak(1,1),ak(1,2))
          if (r.ne.0.0) then
            theta=atan2(y,x)
            do 10 i=1,2
              t(i)=ak(1,i)
              if (ak(2,i).eq.1) then
                t(i)=t(i)*exp(-z)*(cos(z)+sin(z))
              elseif (ak(3,i).eq.1) then
                t(i)=-t(i)*2.0d0*sp*exp(-z)*sin(z)
              elseif (ak(4,i).eq.1) then
                t(i)=t(i)*2.0d0*sp*sp*exp(-z)*(sin(z)-cos(z))
              endif
              t(i)=t(i)*sin(theta)**ak(5,i)
     &             *cos(theta)**ak(6,i)*r**ak(7,i)
10          continue
            lsh=t(1)+t(2)
          else
            do 20 i=1,2
              t(i)=ak(1,i)
              if (ak(3,i).eq.1.and.ak(7,i).eq.0) then
                t(i)=0.0d0
              elseif (ak(3,i).eq.1.and.ak(7,i).eq.-1) then
```

```
              t(i)=-t(i)*2.0d0*sp*sp
            elseif (ak(4,i).eq.1) then
              t(i)=-t(i)*2.0d0*sp*sp
            endif
20        continue
          lsh=t(1)+t(2)
        endif
        return
        end

C-------------------------------------------------------------
C
C  Subroutine DEFSHP returns the parameters to define the
C  required mode shape derivative.

        subroutine defshp(ix,iy,a1,a2)
        integer*1 a1,a2,lookup,table
        dimension a1(7),a2(7)
        dimension lookup(0:2,0:2),table(14,6)
        data table/1,1,12*0,
     &          1,0,1,0,0,1,8*0,
     &          1,0,1,0,1,9*0,
     &          1,0,0,1,0,2,0,1,0,1,0,2,0,-1,
     &          1,0,0,1,2,0,0,1,0,1,0,0,2,-1,
     &          1,0,0,1,1,1,0,-1,0,1,0,1,1,-1/
        data lookup/1,2,4,3,6,0,5,0,0/

C  These are the powers of the terms

        do 10 i=1,7
          a1(i)=table(i,lookup(ix,iy))
          a2(i)=table(i+7,lookup(ix,iy))
10      continue
        return
        end
```

```
C  FILE:  LISANR.FOR
C  These routines are from NUMERICAL RECIPES
C  Copyright Numerical Recipes Software, 1986
      real*8 function brent(ax,bx,cx,f,tol,xmin)
      implicit real*8 (a-h,o-z)
      parameter (itmax=200,cgold=.3819660,zeps=1.0e-10)
      a=min(ax,cx)
      b=max(ax,cx)
      v=bx
      w=v
      x=v
      e=0.
      fx=f(x)
      fv=fx
      fw=fx
      do 11 iter=1,itmax
        xm=0.5*(a+b)
        tol1=tol*abs(x)+zeps
        tol2=2.*tol1
        if(abs(x-xm).le.(tol2-.5*(b-a))) goto 3
        if(abs(e).gt.tol1) then
          r=(x-w)*(fx-fv)
          q=(x-v)*(fx-fw)
          p=(x-v)*q-(x-w)*r
          q=2.*(q-r)
          if(q.gt.0.) p=-p
          q=abs(q)
          etemp=e
          e=d
          if(abs(p).ge.abs(.5*q*etemp).or.p.le.q*(a-x).or.
     *        p.ge.q*(b-x)) goto 1
          d=p/q
          u=x+d
          if(u-a.lt.tol2 .or. b-u.lt.tol2) d=sign(tol1,xm-x)
          goto 2
        endif
1       if(x.ge.xm) then
          e=a-x
        else
          e=b-x
        endif
        d=cgold*e
2       if(abs(d).ge.tol1) then
          u=x+d
        else
          u=x+sign(tol1,d)
        endif
        fu=f(u)
        if(fu.le.fx) then
          if(u.ge.x) then
            a=x
          else
            b=x
          endif
```

```
          v=w
          fv=fw
          w=x
          fw=fx
          x=u
          fx=fu
        else
          if(u.lt.x) then
            a=u
          else
            b=u
          endif
          if(fu.le.fw .or. w.eq.x) then
            v=w
            fv=fw
            w=u
            fw=fu
          else if(fu.le.fv .or. v.eq.x .or. v.eq.w) then
            v=u
            fv=fu
          endif
        endif
11    continue
      pause 'brent exceed maximum iterations.'
3     xmin=x
      brent=fx
      return
      end

      subroutine mnbrak(ax,bx,cx,fa,fb,fc,func)
      implicit real*8 (a-h,o-z)
      parameter (gold=1.618034, glimit=100., tiny=1.e-20)
      fa=func(ax)
      fb=func(bx)
      if(fb.gt.fa)then
        dum=ax
        ax=bx
        bx=dum
        dum=fb
        fb=fa
        fa=dum
      endif
      cx=bx+gold*(bx-ax)
      fc=func(cx)
1     if(fb.ge.fc)then
        r=(bx-ax)*(fb-fc)
        q=(bx-cx)*(fb-fa)
        u=bx-((bx-cx)*q-(bx-ax)*r)/
     *     (2.*sign(max(abs(q-r),tiny),q-r))
        ulim=bx+glimit*(cx-bx)
        if((bx-u)*(u-cx).gt.0.)then
          fu=func(u)
          if(fu.lt.fc)then
            ax=bx
```

```
            fa=fb
            bx=u
            fb=fu
            go to 1
          else if(fu.gt.fb)then
            cx=u
            fc=fu
            go to 1
          endif
          u=cx+gold*(cx-bx)
          fu=func(u)
        else if((cx-u)*(u-ulim).gt.0.)then
          fu=func(u)
          if(fu.lt.fc)then
            bx=cx
            cx=u
            u=cx+gold*(cx-bx)
            fb=fc
            fc=fu
            fu=func(u)
          endif
        else if((u-ulim)*(ulim-cx).ge.0.)then
          u=ulim
          fu=func(u)
        else
          u=cx+gold*(cx-bx)
          fu=func(u)
        endif
        ax=bx
        bx=cx
        cx=u
        fa=fb
        fb=fc
        fc=fu
        go to 1
      endif
      return
      end

      subroutine amoeba(p,y,mp,np,ndim,ftol,funk,iter)
      implicit real*8 (a-h,o-z)
      parameter (nmax=2,alpha=1.0,beta=0.5,
     *  gamma=2.0,itmax=500)
      dimension p(mp,np),y(mp),pr(nmax),prr(nmax),pbar(nmax)
      mpts=ndim+1
      iter=0
1     ilo=1
      if(y(1).gt.y(2))then
        ihi=1
        inhi=2
      else
        ihi=2
        inhi=1
      endif
```

```
      do 11 i=1,mpts
        if(y(i).lt.y(ilo)) ilo=i
        if(y(i).gt.y(ihi))then
          inhi=ihi
          ihi=i
        else if(y(i).gt.y(inhi))then
          if(i.ne.ihi) inhi=i
        endif
11    continue
      rtol=2.*abs(y(ihi)-y(ilo))/(abs(y(ihi))+abs(y(ilo)))
      if(rtol.lt.ftol)return
      if(iter.eq.itmax)
     *  pause 'amoeba exceeding maximum iterations.'
      iter=iter+1
      do 12 j=1,ndim
        pbar(j)=0.
12    continue
      do 14 i=1,mpts
        if(i.ne.ihi)then
          do 13 j=1,ndim
            pbar(j)=pbar(j)+p(i,j)
13        continue
        endif
14    continue
      do 15 j=1,ndim
        pbar(j)=pbar(j)/ndim
        pr(j)=(1.+alpha)*pbar(j)-alpha*p(ihi,j)
15    continue
      ypr=funk(pr)
      if(ypr.le.y(ilo))then
        do 16 j=1,ndim
          prr(j)=gamma*pr(j)+(1.-gamma)*pbar(j)
16      continue
        yprr=funk(prr)
        if(yprr.lt.y(ilo))then
          do 17 j=1,ndim
            p(ihi,j)=prr(j)
17        continue
          y(ihi)=yprr
        else
          do 18 j=1,ndim
            p(ihi,j)=pr(j)
18        continue
          y(ihi)=ypr
        endif
      else if(ypr.ge.y(inhi))then
        if(ypr.lt.y(ihi))then
          do 19 j=1,ndim
            p(ihi,j)=pr(j)
19        continue
          y(ihi)=ypr
        endif
        do 21 j=1,ndim
          prr(j)=beta*p(ihi,j)+(1.-beta)*pbar(j)
```

```
21      continue
        yprr=funk(prr)
        if(yprr.lt.y(ihi))then
          do 22 j=1,ndim
            p(ihi,j)=prr(j)
22        continue
          y(ihi)=yprr
        else
          do 24 i=1,mpts
            if(i.ne.ilo)then
              do 23 j=1,ndim
                pr(j)=0.5*(p(i,j)+p(ilo,j))
                p(i,j)=pr(j)
23            continue
              y(i)=funk(pr)
            endif
24        continue
        endif
      else
        do 25 j=1,ndim
          p(ihi,j)=pr(j)
25      continue
        y(ihi)=ypr
      endif
      go to 1
      end
```

```
C  File:  INTGL.FOR
        program integral
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer bcx,bcy,pi,pj
        real*8 intgl

        write (*,10)
10      format (/'  -- IMPACT MODE INTEGRAL GENERATION --')
        write (*,*) 'Generating mode shape integrals . . .'
        write (*,*)
        open (unit=3,file='intgl.tbl')
        call numint
        close (unit=3)
        end

C-------------------------------------------------------------
C
C  Subroutine NUMINT performs numeric integrations to find
C  the values for the INTGL matrix.

        subroutine numint
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        real*8 intgl
        integer nx,ny,mx,my,pi,pj
        dimension im(4),ix(4),iy(4)
        dimension idx(6),idy(6),ci(0:8)
        dimension intgl(6,2,6,2,0:8)
        dimension supint(5,0:8)
        data pi /3.14159265358979323846/
        data idx /0,1,0,2,0,1/
        data idy /0,0,1,0,2,1/

C  Fill the array with -7's to show that this particular
C  entry has not yet been calculated.  This allows certain
C  symmetries to be exploited later.
C  -7 was chosen as random.  Its choice means that any
C  values which actually equal -7 will be calculated more
C  often than necessary.

        do 5 i=1,6
        do 5 pi=1,2
        do 5 j=1,6
        do 5 pj=1,2
          intgl(i,pi,j,pj,0)=-7.0
5       continue

        itot=144
        do 70 i=1,6
        do 70 pi=1,2
        do 70 j=1,i
        do 70 pj=1,2
          icount=icount+1
```

```
            if (intgl(i,pi,j,pj,0).eq.-7.0) then
              ix(1)=idx(i)
              iy(1)=idy(i)
              if (pi.eq.2) then
                ix(2)=idx(i)
                iy(2)=idy(i)
              else
                ix(2)=-1
              endif
              ix(3)=idx(j)
              iy(3)=idy(j)
              if (pj.eq.2) then
                ix(4)=idx(j)
                iy(4)=idy(j)
              else
                ix(4)=-1
              endif
              call doint(ix,iy,ci)
              do 30 k=0,8
                intgl(i,pi,j,pj,k)=ci(k)
                intgl(j,pj,i,pi,k)=ci(k)
30            continue
              write (*,40) icount,itot,ci(0),ci(1)
40            format ('+',i6,'/',i6,3g16.8)
            endif
          write (3,*) (intgl(i,pi,j,pj,k),k=0,8)
70        continue

            ix(1)=1
            iy(1)=1
            ix(2)=2
            iy(2)=0
            ix(3)=1
            iy(3)=1
            ix(4)=-1
            call doint(ix,iy,ci)
            write (3,*) (ci(k),k=0,8)

            ix(1)=1
            iy(1)=1
            ix(2)=1
            iy(2)=0
            ix(3)=0
            iy(3)=1
            ix(4)=-1
            call doint(ix,iy,ci)
            write (3,*) (ci(k),k=0,8)

            ix(1)=2
            iy(1)=0
            ix(2)=1
            iy(2)=1
            ix(3)=2
            iy(3)=0
```

```
          ix(4)=1
          iy(4)=1
          call doint(ix,iy,ci)
          write (3,*) (ci(k),k=0,8)

          ix(1)=2
          iy(1)=0
          ix(2)=1
          iy(2)=1
          ix(3)=1
          iy(3)=0
          ix(4)=0
          iy(4)=1
          call doint(ix,iy,ci)
          write (3,*) (ci(k),k=0,8)

          ix(1)=1
          iy(1)=0
          ix(2)=0
          iy(2)=1
          ix(3)=1
          iy(3)=0
          ix(4)=0
          iy(4)=1
          call doint(ix,iy,ci)
          write (3,*) (ci(k),k=0,8)
        return
        end

C-------------------------------------------------------------
C
C  Subroutine DOINT returns the values of the integral
C  calculated by reducing it to one dimension and then
C  doing it numerically.
C  Separate terms are returned to be multiplied by various
C  powers of the shape parameter c.

        subroutine doint(ix,iy,ci)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer*1 at
        dimension ix(4),iy(4),ci(0:8)
        dimension weight(0:4,0:4)
        common /msder/ mn,at(7,64)
        external midpnt,midexp,w
        data weight /2,1,0.75,0.625,0.546875,
     &    1,0.25,0.125,0.078125,0.0546875,
     &    0.75,0.125,0.046875,0.0234375,0.01367175,
     &    0.625,0.078125,0.0234375,0.009765625,0.0048828125,
     &    0.546875,0.0546875,0.013671875,
     &    0.0048828125,0.0021362305/
        data pi /3.14159265358979323846/

C  If any of the terms are greater than d2w/dxdy forget it.
```

```
        do 5 k=0,8
          ci(k)=0.0
5       continue

        do 10 i=1,4
          if (ix(i)+iy(i).gt.2) then
            return
          endif
10      continue

        call defshp(ix(1),iy(1),at(1,1),at(1,2))
        istart=1
        iend=2
        do 30 n=2,4
          if (ix(n).ne.-1) then
            if (istart.lt.3) then
              jstart=iend+1
            else
              jstart=1
            endif
            jend=jstart+1
            call defshp(ix(n),iy(n),at(1,jstart),at(1,jend))
            call mult(istart,iend,jstart,jend,at)
          endif
30      continue
        call dothet(istart,iend,at)

        if (iend.lt.istart) return
        do 50 j=0,8
          t=0.0d0
          t1=t
          t2=0.5d0
          t3=1.0d30
          do 40 i=istart,iend
            mn=i
            if (2*at(4,i)+at(3,i).eq.j) then
              call qromo(w,t1,t2,t4,midpnt)
              call qromo(w,t2,t3,t5,midexp)
              t=t+(t4+t5)*weight(at(5,i)/2,at(6,i)/2)
            endif
40        continue
          ci(j)=t*pi
50      continue
        return
        end

C-------------------------------------------------------------
C
C  Function W is the current term of the integral.  It uses
C  the information processed by DOINT.

        real*8 function w(r)
        implicit real*8 (a-h,o-z)
```

```
$include:'spad.inc'
        integer*1 at
        common /msder/ mn,at(7,64)

        t=at(1,mn)*r**at(7,mn)
        if (at(2,mn).ne.0) then
          t=t*(exp(-r)*(cos(r)+sin(r)))**at(2,mn)
        endif
        if (at(3,mn).ne.0) then
          t=t*2.0d0*(-exp(-r)*sin(r))**at(3,mn)
        endif
        if (at(4,mn).ne.0) then
          t=t*2.0d0*(exp(-r)*(-cos(r)+sin(r)))**at(4,mn)
        endif
        w=t
        return
        end

C-------------------------------------------------------------
C
C  Subroutine DEFSHP returns the parameters to define the
C  required mode shape derivative.

        subroutine defshp(ix,iy,a1,a2)
        integer*1 a1,a2,lookup,table
        dimension a1(7),a2(7)
        dimension lookup(0:2,0:2),table(14,6)
        data table/1,1,12*0,
     &          1,0,1,0,0,1,8*0,
     &          1,0,1,0,1,9*0,
     &          1,0,0,1,0,2,0,1,0,1,0,2,0,-1,
     &          1,0,0,1,2,0,0,1,0,1,0,0,2,-1,
     &          1,0,0,1,1,1,0,-1,0,1,0,1,1,-1/
        data lookup/1,2,4,3,6,0,5,0,0/

C  These are the powers of the terms

        do 10 i=1,7
          a1(i)=table(i,lookup(ix,iy))
          a2(i)=table(i+7,lookup(ix,iy))
10      continue
        return
        end

C-------------------------------------------------------------
C
C  Subroutine MULT multiplies two terms together
C  The first terms are at(i1-i2), and the second at(j1-j2)

        subroutine mult(i1,i2,j1,j2,at)
        integer*1 at
        dimension at(7,64)

        nterm=(i2-i1+1)*(j2-j1+1)
```

```
          first=min(i1,j1)
          last=max(i2,j2)
          inter=max(j1-i2-1,i2-j1-1)
          if (first.gt.nterm) then
            kstart=1
          elseif (inter.ge.nterm) then
            kstart=min(i2+1,j2+1)
          else
            kstart=last+1
          endif
          kend=kstart+nterm-1
          do 30 i=i1,i2
            do 20 j=j1,j2
              n=(i-i1)*(j2-j1+1)+j-j1+1
              at(1,kstart+n-1)=at(1,i)*at(1,j)
              do 10 k=2,7
                at(k,kstart+n-1)=at(k,i)+at(k,j)
10            continue
20          continue
30        continue
          i1=kstart
          i2=kend
          return
          end

C-------------------------------------------------------------
C
C  Subroutine DOTHET eliminates integral which evaluate to
C  zero in theta.  It then multiplies all terms by r and
C  collects like terms.

          subroutine dothet(i1,i2,at)
          integer*1 at,index
          dimension at(7,64)
          dimension index(64)

C  Loop over all terms

          do 10 i=i1,i2

C  Add one to the r exponent

            at(7,i)=at(7,i)+1

C  Are both the sin and cos exponent even?

            if (mod(at(5,i),2)+mod(at(6,i),2).ne.0) then

C  No, so zap the coefficient.

              at(1,i)=0
            endif
10        continue
```

```
        n=1
        if (at(1,i1).ne.0) then
          index(n)=i1
          n=n+1
        endif
        do 100 i=i1+1,i2
          f=1
          if (at(1,i).ne.0) then
            f=0
            do 90 j=i1,i-1
              t=0
              do 20 k=2,7
                if (at(k,i).ne.at(k,j)) t=1
20            continue
              if (t.eq.0) then
                at(1,i)=at(1,i)+at(1,j)
               f=1
              endif
90          continue
          endif
          if (f.eq.0) then
            index(n)=i
            n=n+1
          endif
100     continue
        do 120 i=1,n-1
          do 110 j=1,7
            at(j,i)=at(j,index(i))
110       continue
120     continue
        i1=1
        i2=n-1
        return
        end
```

```
C  File:  QROMB.FOR
C  These routines are directly from NUMERICAL RECIPES
C  Copyright Numerical Recipes Software, 1986
      subroutine qromo(func,a,b,ss,choose)
      implicit real*8 (a-h,o-z)
      external func,choose
      parameter (eps=1.d-8, jmax=14, jmaxp=jmax+1, k=5,
&       km=k-1)
      dimension s(jmaxp),h(jmaxp)
      zero=0.0d0
      h(1)=1.
      do 11 j=1,jmax
        call choose(func,a,b,s(j),j)
        if (j.ge.k) then
          call polint(h(j-km),s(j-km),k,zero,ss,dss)
          if (abs(dss).lt.eps*abs(ss)) return
        endif
        s(j+1)=s(j)
        h(j+1)=h(j)/9.
11    continue
      stop 'Too many steps in QROMO.'
      end

      subroutine midpnt(func,a,b,s,n)
      implicit real*8 (a-h,o-z)
      external func
      if (n.eq.1) then
        s=(b-a)*func(0.5*(a+b))
        it=1
      else
        tnm=it
        del=(b-a)/(3.*tnm)
        ddel=del+del
        x=a+0.5*del
        sum=0.
        do 11 j=1,it
          sum=sum+func(x)
          x=x+ddel
          sum=sum+func(x)
          x=x+del
11      continue
        s=(s+(b-a)*sum/tnm)/3.
        it=3*it
      endif
      return
      end

      subroutine midexp(funk,aa,bb,s,n)
      implicit real*8 (a-h,o-z)
      external funk
      func(x)=funk(1.0d0/x)/(x*x)
      b=1.0d0/aa
      a=0.0d0
      if (n.eq.1) then
```

```
        s=(b-a)*func(0.5*(a+b))
        it=1
      else
        tnm=it
        del=(b-a)/(3.*tnm)
        ddel=del+del
        x=a+0.5*del
        sum=0.
        do 11 j=1,it
          sum=sum+func(x)
          x=x+ddel
          sum=sum+func(x)
          x=x+del
11      continue
        s=(s+(b-a)*sum/tnm)/3.
        it=3*it
      endif
      return
      end

      subroutine polint(xa,ya,n,x,y,dy)
      implicit real*8 (a-h,o-z)
      parameter (nmax=10)
      dimension xa(n),ya(n),c(nmax),d(nmax)
      ns=1
      dif=abs(x-xa(1))
      do 11 i=1,n
        dift=abs(x-xa(i))
        if (dift.lt.dif) then
          ns=i
          dif=dift
        endif
        c(i)=ya(i)
        d(i)=ya(i)
11    continue
      y=ya(ns)
      ns=ns-1
      do 13 m=1,n-1
        do 12 i=1,n-m
          ho=xa(i)-x
          hp=xa(i+m)-x
          w=c(i+1)-d(i)
          den=ho-hp
          if (den.eq.0.) stop 'Error in POLINT'
          den=w/den
          d(i)=hp*den
          c(i)=ho*den
12      continue
        if (2*ns.lt.n-m)then
          dy=c(ns+1)
        else
          dy=d(ns)
          ns=ns-1
        endif
```

```
        y=y+dy
13      continue
        return
        end
```

```
C  File:  SPAD.FOR
$LARGE:Mass,K,MK
      program spad
C  Program SPAD does the actual analysis of the
C  impact event described by the input file (no extension).

C  Set up variables and array space:

      implicit real*8 (a-h,o-z)
      real*8 Mass,K,MK
$include:'spad.inc'
      external f
      integer bcx,bcy,c
      dimension q0(0:1),q0v(0:1),q0a(0:1)
      dimension q(maxdim,0:1),qv(maxdim,0:1),qa(maxdim,0:1)
      dimension Qf(maxdim),w(maxdim),qt(maxdim)
      dimension v1(maxdim),v3(maxdim),tq(maxdim)
      dimension index(maxdim)
      dimension Mass(maxvec+1),K(maxvec)
      dimension MK(maxdim,maxdim)
      equivalence (Mass(maxvec+1),K(1)),(Mass(1),MK(1,1))

C  Get input file name and read input data:

      write (*,10)
10    format (/' -- Sandwich Panel Analysis of Damage --'//)
      write (*,*) 'Reading input file . . .'
      call input(bcx,bcy)

C  Generate the integrals of beam functions for use in
C  determining
C  the mass and stiffness matrices:

      write (*,*) 'Calculating mode shape integrals . . .'
      call ibfunc(bcx,bcy)

C  Generate and invert the mass matrix:
C  Generate the stiffness matrix:

      write (*,*) 'Generating mass/stiffness matrices . . .'
      call genm(Mass,index)
      call genk(K,index)

C  Now find q1 and q4 locations:

      i=1
20    if (index(i).eq.0) then
        i=i+1
        goto 20
      endif
      q1beg=index(i)
      i=nv
25    if (index(i).eq.0) then
        i=i-1
```

```
        goto 25
      endif
      q1end=index(i)
      i=nv*3+1
30    if (index(i).eq.0) then
        i=i+1
        goto 30
      endif
      q4beg=index(i)
      i=nv*4
35    if (index(i).eq.0) then
        i=i-1
        goto 35
      endif
      q4end=index(i)

C  Report rank of resulting problem:

      write (*,40) rank,nv*6
40    format (' Rank:',i4,'/',i4)

C  Now we calculate the {Q} vector

c     write (*,*) 'Generating the Q vector . . .'
      call genq(Qf,index)

C  Now compute [A1]=[4/dt/dt*Mass+K]-1 and put
C  the result in [K]:

      write (*,*) 'Generating the A1 matrix . . .'
      call gena1(Mass,K)

C  And the {W} vector which relates center displacement to
C     the generalized coordinates

c     write (*,*) 'Generating the W vector . . .'
      call genw(w,index)

C  The quantities {v1}=[A1]*{Q}
C                 {v3}t={W}t*[A1]*[Mass]
C           and  [MK]=[A1]*[Mass]
C   will be useful so compute them now

      write (*,*) 'Generating internal vectors . . .'
      call genv(w,K,Mass,MK,Qf,v1,v3)

C  The factor ha=-dt*dt/4/m-{W}t*[A1]*{Q} need only be
C  calculated once.

      call genha(w,v1,ha)

C  We now have all necessary information, so we can solve
C  the problem
```

```
C  Set up the initial conditions:

      c=0
      n=1
      f1=4.0/dt/dt
      force=0.0
      fmax=0.0
      q0(c)=0.0
      q0v(c)=v
      q0a(c)=0.0
      do 70 i=1,rank
        q(i,c)=0.0
        qv(i,c)=0.0
        qa(i,c)=0.0
        tq(i)=0.0
70    continue
      tol=1e-8
      time=0.0
      niter=1
      iter=0

      call filext(fname,'.frc')
      open(unit=3,file=fname)

      call filext(fname,'.wa ')
      open(unit=4,file=fname)

      call filext(fname,'.wb ')
      open(unit=5,file=fname)

      call filext(fname,'.q0 ')
      open(unit=6,file=fname)

      write (*,80)
80    format (' Time           q0              wa'
     &          '              wb          Force')

      wa=0.0
      do 85 i=q1beg,q1end
        wa=wa+q(i,c)*w(i)
85    continue

      wb=0.0
      do 88 i=q4beg,q4end
        wb=wb+q(i,c)*w(i-q4beg+q1beg)
88    continue

      write (*,90) time,q0(c),wa,wb,force
      write (3,95) time,force
      write (4,95) time,wa
      write (5,95) time,wb
      write (6,95) time,q0(c)

90    format (1x,5g15.8)
```

```
95   format (1x,2g15.8)

C////////////////////////////////////////////////////////////
C
C  This is the top of the time integration loop.
C  The method used here is Newmark constant-average
C  acceleration method.

C  Step 1.
C  Calculate
hb=q0+qv0*dt+qa0*dt*dt/4-{v3}t*{qa+f1*(q+qv*dt)}

100  t=0.0
     do 110 i=1,rank
       qt(i)=qa(i,c)+f1*(q(i,c)+qv(i,c)*dt)
       t=t+v3(i)*qt(i)
110  continue
     hb=q0(c)+(q0v(c)+q0a(c)*dt/4)*dt-t

C  Step 2.  Solve for f(c) using Newton/Raphson

     call solvef(force,ha,hb,hk,tol)
     if (force.lt.0) force=0.0

C  Step 3.  Calculate q(n)

     do 150 m=1,rank
       t=0.0
       do 130 j=1,rank
         t=t+MK(m,j)*qt(j)
130    continue
       q(m,n)=v1(m)*force+t
150  continue

C  Step 4.  Calculate qa0(n)

     q0a(n)=-force/mi

C  Step 5.  Calculate qv0(n) and q0(n)

     q0v(n)=q0v(c)+dt/2*(q0a(c)+q0a(n))
     q0(n)=q0(c)+dt/2*(q0v(c)+q0v(n))

C  Step 6.  Calculate qv(n) and qa(n)

     do 160 i=1,rank
       qv(i,n)=2/dt*(q(i,n)-q(i,c))-qv(i,c)
       qa(i,n)=2/dt*(qv(i,n)-qv(i,c))-qa(i,c)
160  continue

C  Save variables at maximum force

     if (force.gt.fmax) then
       fmax=force
```

```
        do 165 i=1,rank
          tq(i)=q(i,n)
165     continue
      endif

C  That's all -- Report Results

      time=time+dt

      iter=iter+1
      if (iter.ge.niter) then
        iter=0
        wa=0.0
        do 170 i=q1beg,q1end
          wa=wa+q(i,n)*w(i)
170     continue
        wb=0.0
        do 180 i=q4beg,q4end
          wb=wb+q(i,n)*w(i-q4beg+q1beg)
180     continue

        write (*,90) time,q0(n),wa,wb,force
        write (3,95) time,force
        write (4,95) time,wa
        write (5,95) time,wb
        write (6,95) time,q0(c)
      endif

C  Update the time registers . . .

      c=n
      n=1-n

C  . . . and loop

      if (q0(n).ge.0.0.or.force.gt.0.0) goto 100

C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\


      close (unit=3)
      close (unit=4)
      close (unit=5)
      close (unit=6)

      call filext(fname,'.dis')
      open (unit=3,file=fname)
      ndiv=50
      xmid=0.5
      xinc=1.0/ndiv
      do 210 i=0,ndiv
        x=i*xinc
        wa=0.0
        j=0
```

```
        do 200 iy=1,nm(2)
        do 200 ix=1,nm(1)
          if (index(idof(ix,iy,1)).ne.0) then
            j=j+1
            wa=wa+tq(j)*shape(1,ix,0,x)*shape(2,iy,0,xmid)
          endif
200     continue
        write (3,95) x,wa
210   continue
      close (unit=3)

      call filext(fname,'.tq ')
      open (unit=3,file=fname)
      write (3,*) (fmax/hk)**(2.0/3.0),rank
      write (3,*) nm(1),nm(2)
      do 220 iv=1,6
      do 220 ix=1,nm(1)
      do 220 iy=1,nm(2)
        if (index(idof(ix,iy,iv)).ne.0) then
          write (3,*) ix,iy,iv,tq(index(idof(ix,iy,iv)))
        endif
220   continue
      do 230 i=1,2
        write (3,*) (bcbeta(j,i),j=1,nm(i))
        write (3,*) bcthet(i),bca(i)
        write (3,*) (bcb(j,i),j=1,nm(i))
230   continue
      close (unit=3)

9000  end

C-------------------------------------------------------------
C
C  Subroutine SOVLEF finds the value of x which satisfies:
C    h*(a*x+b)^1.5-x=0 by the Newton/Raphson method
C
      subroutine solvef (x,a,b,h,tol)
      implicit real*8 (a-h,o-z)

10    alpha=a*x+b
      if (alpha.le.0.0) then
        x=0.0
        return
      endif
      f=h*(alpha)**1.5-x
      if (abs(f).le.tol) return
      x=x-f/(1.5*h*a*sqrt(alpha)-1.0)
      goto 10
      end

C-------------------------------------------------------------
C
C  Subroutine GENQ calculates the vector {Q}
C  less the constant equal to the total force.
```

```
      subroutine genq(Qf,index)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'
      external f
      dimension index(maxdim),Qf(maxdim)

      do 10 m=1,nm(1)
      do 10 n=1,nm(2)
        i=(n-1)*nm(1)+m
        Qf(i)=smpsnr(f,m,n,10,10,al,ah,bl,bh)
        do 10 j=1,5
          Qf(i+j*nv)=0.0
10    continue

C  Trash unused variables

      do 20 i=1,6*nv
        if (index(i).ne.0) Qf(index(i))=Qf(i)
20    continue

      return
      end

C-------------------------------------------------------------

C
C  Subroutine GENW calculates center displacements.  Only
values
C  associated with q1 are needed or computed.

      subroutine genw(w,index)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'
      dimension w(maxdim),index(maxdim)

      t=0.5
      do 10 m=1,nm(1)
      do 10 n=1,nm(2)
        i=(n-1)*nm(1)+m
        w(i)=shape(1,m,0,t)*shape(2,n,0,t)
        do 10 j=1,5
          w(i+j*nv)=0.0
10    continue

C  Again, trash unused variables

      do 20 i=1,nv*6
        if (index(i).ne.0) w(index(i))=w(i)
20    continue
      return
      end

C-------------------------------------------------------------
```

```
C
C  Subroutine GENV calculates the following quantities:
C    {v1}=[A1]*{Q}
C    {v2}t={W}t*[A1]
C       {v3}t={v2}t*[Mass]
C   and [MK] = [A1]*[Mass]
C  for use in later calculations

      subroutine genv(w,K,Mass,MK,Qf,v1,v3)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'
      real*8 K,Mass,MK
      dimension K(maxvec),Mass(maxvec),MK(maxdim,maxdim)
      dimension w(maxdim),Qf(maxdim)
      dimension v1(maxdim),v2(maxdim),v3(maxdim)

      do 20 i=1,rank
        t1=0.0
        t2=0.0
        do 10 j=1,rank
          t1=t1+K(indp(i,j))*Qf(j)
          t2=t2+w(j)*K(indp(i,j))
10      continue
        v1(i)=t1
        v2(i)=t2
20    continue

      do 40 i=1,rank
        t1=0.0
        do 30 j=1,rank
          t1=t1+v2(j)*Mass(indp(i,j))
30      continue
        v3(i)=t1
40    continue

C  MK will overwrite both Mass and K, so store the result to
a
C  temporary file while we finish calculations

      open(unit=2,form='binary',status='scratch')
      do 70 i=1,rank
      do 60 j=1,rank
        t=0.0
        do 50 m=1,rank
          t=t+K(indp(i,m))*Mass(indp(m,j))
50      continue
        v2(j)=t
60    continue
      write (2) (v2(j),j=1,rank)
70    continue
      rewind(unit=2)
      do 80 i=1,rank
        read (2) (MK(i,j),j=1,rank)
80    continue
```

```
      close(unit=2,status='delete')
      return
      end

C-------------------------------------------------------------
C
C  Subroutine GENHA calculates the quantity:
C     -dt*dt/4/m-{W}*[4/dt/dt*Mass+K]-1*{Q}
C  for use in later calculations

      subroutine genha(w,v1,ha)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'
      dimension w(maxdim),v1(maxdim)

      t=0.0
      do 10 i=1,rank
        t=t+w(i)*v1(i)
10    continue
      ha=-dt*dt/4/mi-t
      return
      end

C-------------------------------------------------------------
C
C  Function SMPSNR performs an integration of
C  the function f (x,y)
C  on the specified interval within [0:1].
C
      real*8 function smpsnr(f,m,n,ipx,ipy,xl,xh,yl,yh)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'
      external f
      xint=(xh-xl)/ipx
      yint=(yh-yl)/ipy
      t2=0.0
      do 20 i=1,ipx-1
        x=xl+xint*i
        t=0.0
        do 10 j=1,ipy-1
          y=yl+yint*j
          t=t+(mod(j,2)*2+2)*f(x,y)*shape(2,n,0,y)
10      continue
        t=t+f(x,yl)*shape(2,n,0,yl)+f(x,yh)*shape(2,n,0,yh)
        t2=t2+t*shape(1,m,0,x)*(mod(i,2)*2+2)
20    continue

      t=0.0
      do 30 j=1,ipy-1
        y=yl+yint*j
        t=t+(mod(j,2)*2+2)*f(xl,y)*shape(2,n,0,y)
30    continue
      t=t+f(xl,yl)*shape(2,n,0,yl)+f(xl,yh)*shape(2,n,0,yh)
      t2=t2+t*shape(1,m,0,xl)
```

```
      t=0.0
      do 40 j=1,ipy-1
        y=yl+yint*j
        t=t+(mod(j,2)*2+2)*f(xh,y)*shape(2,n,0,y)
40    continue
      t=t+f(xh,yl)*shape(2,n,0,yl)+f(xh,yh)*shape(2,n,0,yh)
      t2=t2+t*shape(1,m,0,xh)

      smpsnr=t2*xint/3*yint/3
      return
      end

C-------------------------------------------------------------
C
C  Function SHAPE returns the value of:
C    the pth derivative of mode shape n in the i direction
C    at point x.

      real*8 function shape(i,n,p,x)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'
      integer i,n,p

      r2=sqrt(2.0)
      beta=bcbeta(n,i)
      if (beta.eq.0) then
        if (p.eq.0) then
          shape=1.0
        else
          shape=0.0
        endif
      else
        theta=beta*x+bcthet(i)
        e1=bca(i)*exp(-beta*x)
        e2=bcb(n,i)*exp(-beta*(1.0-x))
        if (p.eq.0) then
          shape=r2*dsin(theta)+e1+e2
        elseif (p.eq.1) then
          shape=beta*(r2*dcos(theta)-e1+e2)
        else
          shape=beta*beta*(-r2*dsin(theta)+e1+e2)
        endif
      endif
      return
      end


C-------------------------------------------------------------

C
C  Function f(x,y) returns the value of the impact load at
C  point (x,y) on the plate.  It is scaled such that the
C  total load is 1.0
```

```
      real*8 function f(x,y)
      implicit real*8 (a-h,o-z)
$include:'spad.inc'

      if (x.ge.al.and.x.le.ah.and.y.ge.bl.and.y.le.bh) then
         f=1.0/((ah-al)*(bh-bl))
      else
         f=0.0
      endif
      return
      end
```

```
C  File:  SPADRT.FOR
C-------------------------------------------------------------
C
C  Subroutine FILEXT appends the extension ext to name.
C
        subroutine filext(name,ext)
        character*40 name
        character*4 ext
        j=0
        do 10 i=1,36
          if (name(i:i).eq.'.'
     &         .or.name(i:i).eq.' '.and.j.eq.0) j=i
10      continue
        if (j.eq.0) then
          write (*,*) 'File name too long.'
          stop
        endif
        name(j:j+3)=ext(1:4)
        name(j+4:j+4)=' '
        return
        end

C-------------------------------------------------------------
C
C  Subroutine INPUT reads data from a prepared file for use
C  by the analysis program.
C  These data files are created by PROBLEM.FOR and have
C  no extension.

        subroutine input(bcx,bcy)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer bcx,bcy
        dimension diam(4)
        character*40 line
        pi=3.14159265358979323846

        open(unit=2,file=' ',err=110)
        read (2,10) line
10      format (a40)
        j=1
20      j=j+1
        if (line(j:j).eq.' '.or.line(j:j).eq.09) goto 20
        k=j
30      k=k+1
        if (line(k:k).ne.' ') goto 30
        line(k:k)='.'
        fname(1:k-j+1)=line(j:k)
        read (2,*,err=100) a,b,bcx,bcy
        read (2,*,err=100) (aa(i),i=1,6)
        read (2,*,err=100) ta,rhoa
        read (2,*,err=100) (ab(i),i=1,6)
        read (2,*,err=100) tb,rhob
        read (2,*,err=100) Ezz,Gxz,Gyz,h,rhoc
```

```
        read (2,*,err=100) rhof,tf
        read (2,*,err=100) mi,v,hk,ri
        read (2,*,err=100) nm(1),nm(2)
        read (2,*,err=100) dt
        close (unit=2)
        zac=h/2
        zbc=-zac
        za=zac+ta/2
        zb=zbc-tb/2
        area=a*b
        di=sqrt(pi*ri*ri)
        al=0.5-di/2/a
        ah=0.5+di/2/a
        bl=0.5-di/2/b
        bh=0.5+di/2/b

        nv=nm(1)*nm(2)
        return

100     write (*,105) fname
105     format (' Error reading data from file: ',a40)
        goto 120
110     write (*,115) fname
115     format (' Error opening file: ',a40)
120     stop
        end

C-------------------------------------------------------------
C
C  Subroutine WRMAT writes a packed square matrix of size
C  (rank) to any unit.

        subroutine wrmat(n,am)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        dimension am(maxvec)

        k=0
        do 10 i=1,rank
          write (n,20) (am(k+j),j=1,i)
          k=k+i
10      continue
20      format (1x,6g12.6)
        return
        end

C-------------------------------------------------------------
C
C  Subroutine GENM generates the mass matrix using
C  the beam function integrals psi and phi.


        subroutine genm (Mass,index)
        implicit real*8 (a-h,o-z)
```

```
$include:'spad.inc'
        integer r,s
        real*8 Mass
        dimension Mass(maxvec),index(maxvec)

        a2=a*a
        a3=a2*a
        b2=b*b
        b3=b2*b
        nx=nm(1)
        ny=nm(2)

C  terms from q1mn . . .

        do 10 m=1,nx
        do 10 n=1,ny
          i=(n-1)*nx+m
          do 10 r=1,nx
          do 10 s=1,n
            j=(s-1)*nx+r
            if (i.ge.j) Mass(indp(i,j))=area*
     &        (rhoa*ta+rhof*tf+rhoc*h/3)*
     &        psi(0,0,m,r)*phi(0,0,n,s)
10      continue


C  terms from q2mn . . .

        do 20 m=1,nx
        do 20 n=1,ny
          i=(n-1)*nx+m+nv
          do 20 r=1,nx
          do 20 s=1,n
            j=(s-1)*nx+r+nv
            if (i.ge.j) Mass(indp(i,j))=area*
     &        (rhoa*ta*za*za+(rhof*tf+rhoc*h/3)*zac*zac)*
     &        psi(1,1,m,r)*phi(0,0,n,s)/a2
20      continue


C  terms from q3mn . . .

        do 30 m=1,nx
        do 30 n=1,ny
          i=(n-1)*nx+m+nv*2
          do 30 r=1,nx
          do 30 s=1,n
            j=(s-1)*nx+r+nv*2
            if (i.ge.j) Mass(indp(i,j))=area*
     &        (rhoa*ta*za*za+(rhof*tf+rhoc*h/3)*zac*zac)*
     &        psi(0,0,m,r)*phi(1,1,n,s)/b2
30      continue
```

```
C  terms from q4mn . . .

         do 40 m=1,nx
         do 40 n=1,ny
           i=(n-1)*nx+m+nv*3
           do 40 r=1,nx
           do 40 s=1,ny
             j=(s-1)*nx+r
             if (i.ge.j+nv*3) Mass(indp(i,j+nv*3))=
     &         area*(rhob*tb+rhof*tf+rhoc*h/3)*
     &         psi(0,0,m,r)*phi(0,0,n,s)
             Mass(indp(i,j))=
     &         area*rhoc*h/6*psi(0,0,m,r)*phi(0,0,n,s)
40       continue


C  terms from q5mn . . .

         do 50 m=1,nx
         do 50 n=1,ny
           i=(n-1)*nx+m+nv*4
           do 50 r=1,nx
           do 50 s=1,ny
             j=(s-1)*nx+r
             if (i.ge.j+nv*4) Mass(indp(i,j+nv*4))=area*
     &         (rhob*tb*zb*zb+(rhof*tf+rhoc*h/3)*zbc*zbc)*
     &         psi(1,1,m,r)*phi(0,0,n,s)/a2
             Mass(indp(i,j+nv))=area*
     &         rhoc*h/6*zac*zbc*psi(1,1,m,r)*phi(0,0,n,s)/a2
50       continue


C  terms from q6mn . . .

         do 60 m=1,nx
         do 60 n=1,ny
           i=(n-1)*nx+m+nv*5
           do 60 r=1,nx
           do 60 s=1,ny
             j=(s-1)*nx+r
             if (i.ge.j+nv*5) Mass(indp(i,j+nv*5))=area*
     &         (rhob*tb*zb*zb+(rhof*tf+rhoc*h/3)*zbc*zbc)*
     &         psi(0,0,m,r)*phi(1,1,n,s)/b2
             Mass(indp(i,j+nv*2))=area*
     &         rhoc*h/6*zac*zbc*psi(0,0,m,r)*phi(1,1,n,s)/b2
60       continue


C  In certain cases, such as free-free modes, we will have
C  zeroes on the main diagonal because some displacements
C  are restrained to be zero in these modes.  In this case
C  we must remove these rows and columns from the mass and
C  stiffness matrices.
```

```
C  Setting the index of any d.o.f to zero will cause that
C  mode to be eliminated.

         do 70 i=1,nv*6
           index(i)=1
70       continue

         call pack(Mass,index,0)

         return
         end

C-------------------------------------------------------------
C
C  Subroutine GENK generates the stiffness matrix using
C  the beam function integrals psi and phi.
C

         subroutine genk (K,index)
         implicit real*8 (a-h,o-z)
$include:'spad.inc'
         integer r,s
         real*8 K
         dimension K(maxvec)

         a2=a*a
         a3=a2*a
         b2=b*b
         b3=b2*b
         nx=nm(1)
         ny=nm(2)

C  terms from qi=q1mn . . .

         do 10 m=1,nx
         do 10 n=1,ny
           i=(n-1)*nx+m
           do 10     r=1,nx
           do 10 s=1,n
             j=(s-1)*nx+r
             if (i.ge.j) K(indp(i,j))=area*
     &         (Ezz/h*psi(0,0,m,r)*phi(0,0,n,s)+
     &         Gyz*h/3*psi(0,0,m,r)*phi(1,1,n,s)/b2+
     &         Gxz*h/3*psi(1,1,m,r)*phi(0,0,n,s)/a2)
10       continue

C  terms from qi=q2mn . . .

         do 20 m=1,nx
         do 20 n=1,ny
           i=nv+(n-1)*nx+m
           do 20     r=1,nx
           do 20 s=1,ny
```

```
            j=(s-1)*nx+r
            K(indp(i,j))=area*
     &        Gxz*zac/3*psi(1,1,m,r)*phi(0,0,n,s)/a2
            if (i.ge.j+nv) K(indp(i,j+nv))=area*(za*za*(
     &      aa(1)*psi(2,2,m,r)*phi(0,0,n,s)/(a2*a2)+
     &      aa(4)*(psi(2,1,m,r)*phi(0,1,n,s)+
     &      psi(1,2,m,r)*phi(1,0,n,s))/(a3*b)+
     &      aa(6)*psi(1,1,m,r)*phi(1,1,n,s)/(a2*b2))+
     &      zac*zac*Gxz/h*psi(1,1,m,r)*phi(0,0,n,s)/a2)
20       continue

C  terms from qi=q3mn . . .

         do 30 m=1,nx
         do 30 n=1,ny
           i=nv*2+(n-1)*nx+m
           do 30     r=1,nx
           do 30 s=1,ny
             j=(s-1)*nx+r
             K(indp(i,j))=area*
     &        Gyz*zac/3*psi(0,0,m,r)*phi(1,1,n,s)/b2
             K(indp(i,j+nv))=area*za*za*(
     &      aa(3)*psi(0,2,m,r)*phi(2,0,n,s)/(a2*b2)+
     &      aa(4)*psi(1,2,m,r)*phi(1,0,n,s)/(a3*b)+
     &      aa(5)*psi(1,0,m,r)*phi(1,2,n,s)/(a*b3)+
     &      aa(6)*psi(1,1,m,r)*phi(1,1,n,s)/(a2*b2))
             if (i.ge.j+nv*2) K(indp(i,j+nv*2))=area*(za*za*(
     &      aa(2)*psi(0,0,m,r)*phi(2,2,n,s)/(b2*b2)+
     &      aa(5)*(psi(1,0,m,r)*phi(1,2,n,s)+
     &      psi(0,1,m,r)*psi(2,1,n,s))/(a*b3)+
     &      aa(6)*psi(1,1,m,r)*phi(1,1,n,s))+
     &      zac*zac*Gyz/h*psi(0,0,m,r)*phi(1,1,n,s)/b2)
30       continue

C  terms from qi=q4mn . . .

         do 40 m=1,nx
         do 40 n=1,ny
           i=nv*3+(n-1)*nx+m
           do 40     r=1,nx
           do 40 s=1,ny
             j=(s-1)*nx+r
             K(indp(i,j))=-area*
     &        (Ezz/h*psi(0,0,m,r)*phi(0,0,n,s)+
     &        Gyz*h/6*psi(0,0,m,r)*phi(1,1,n,s)/b2+
     &        Gxz*h/6*psi(1,1,m,r)*phi(0,0,n,s)/a2)
             K(indp(i,j+nv))=-area*Gxz*zac/3*
     &        psi(1,1,m,r)*phi(0,0,n,s)/a2
             K(indp(i,j+nv*2))=-area*Gyz*zac/3*
     &        psi(0,0,m,r)*phi(1,1,n,s)/b2
             if (i.ge.j+nv*3) K(indp(i,j+nv*3))=area*
     &        (Ezz/h*psi(0,0,m,r)*phi(0,0,n,s)+
     &        Gyz*h/3*psi(0,0,m,r)*phi(1,1,n,s)/b2+
     &        Gxz*h/3*psi(1,1,m,r)*phi(0,0,n,s)/a2)
```

```
40      continue

C  terms from qi=q5mn . . .

        do 50 m=1,nx
        do 50 n=1,ny
          i=nv*4+(n-1)*nx+m
          do 50     r=1,nx
          do 50 s=1,ny
            j=(s-1)*nx+r
            K(indp(i,j))=-area*
     &        Gxz/3*zbc*psi(1,1,m,r)*phi(0,0,n,s)/a2
            K(indp(i,j+nv))=-area*Gxz/h*zac*zbc*
     &      psi(1,1,m,r)*phi(0,0,n,s)/a2
            K(indp(i,j+nv*2))=0
            K(indp(i,j+nv*3))=area*
     &        Gxz*zbc/3*psi(1,1,m,r)*phi(0,0,n,s)/a2
            if (i.ge.j+nv*4) K(indp(i,j+nv*4))=area*(zb*zb*(
     &      ab(1)*psi(2,2,m,r)*phi(0,0,n,s)/(a2*a2)+
     &      ab(4)*(psi(2,1,m,r)*phi(0,1,n,s)+
     &      psi(1,2,m,r)*phi(1,0,n,s))/(a3*b)+
     &      ab(6)*psi(1,1,m,r)*phi(1,1,n,s)/(a2*b2))+
     &      zbc*zbc*Gxz/h*psi(1,1,m,r)*phi(0,0,n,s)/a2)
50      continue

C  terms from qi=q6mn . . .

        do 60 m=1,nx
        do 60 n=1,ny
          i=nv*5+(n-1)*nx+m
          do 60 r=1,nx
          do 60 s=1,ny
            j=(s-1)*nx+r
            K(indp(i,j))=-area*
     &        Gyz/3*zbc*psi(0,0,m,r)*phi(1,1,n,s)/b2
            K(indp(i,j+nv))=0
            K(indp(i,j+nv*2))=-area*Gyz/h*zac*zbc*
     &        psi(0,0,m,r)*phi(1,1,n,s)/b2
            K(indp(i,j+nv*3))=area*Gyz*zbc/3*
     &        psi(0,0,m,r)*phi(1,1,n,s)/b2
            K(indp(i,j+nv*4))=area*zb*zb*(
     &        ab(3)*psi(0,2,m,r)*phi(2,0,n,s)/(a2*b2)+
     &        ab(4)*psi(1,2,m,r)*phi(1,0,n,s)/(a3*b)+
     &        ab(5)*psi(1,0,m,r)*phi(1,2,n,s)/(a*b3)+
     &        ab(6)*psi(1,1,m,r)*phi(1,1,n,s)/(a2*b2))
            if (i.ge.j+nv*5) K(indp(i,j+nv*5))=area*(zb*zb*(
     &        ab(2)*psi(0,0,m,r)*phi(2,2,n,s)/(b2*b2)+
     &        ab(5)*(psi(1,0,m,r)*phi(1,2,n,s)+
     &        psi(0,1,m,r)*psi(2,1,n,s))/(a*b3)+
     &        ab(6)*psi(1,1,m,r)*phi(1,1,n,s))+
     &        zbc*zbc*Gyz/h*psi(0,0,m,r)*phi(1,1,n,s)/b2)
60      continue

C  Now remove the rows and columns corresponding to unused
```

```
C  variables.

        call pack(k,index,1)

        return
        end

C-----------------------------------------------------------
C
C  Function INDP returns the correct index for packed arrays
C
        integer function indp(i,j)
        integer i,j

        if (i.ge.j) then
          indp=i*(i-1)/2+j
        else
          indp=j*(j-1)/2+i
        endif
        return
        end

C-----------------------------------------------------------
C
C  Subroutine GENA1 finds A1=[4/dt/dt*Mass+K]-1 and stores
C  the result in [K].
C

        subroutine gena1(Mass,K)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        real*8 Mass,K
        dimension Mass(maxvec),K(maxvec)
        dimension vec(maxdim),z(maxdim),det(2)

C  First form [4/dt/dt*Mass+K]

        t=4.0/dt/dt
        do 10 j=1,rank*(rank+1)/2
            K(j)=t*Mass(j)+K(j)
10      continue

C  Now we use LINPACK routines to invert the matrix.

        call dppco(K,rank,rcond,z,info)
        if (1.0+rcond.eq.1.0) then
          write (*,20) rcond
20        format (' Mass/Stiffness matrix is singular --'/
     &    ' condition: ',g16.8)
          stop
        endif
        call dppdi(K,rank,det,01)
        return
        end
```

```
C-------------------------------------------------------------
C
C  Subroutine PACK removes zeroes from the main diagonal of
C  a matrix.
C
C  On input:
C    job=0   find and remove zeroes and leave record in index
C    job=1   remove zeroes using index
C  On output:
C    index(i)  final index of original row/column i
C               (0 if deleted)

        subroutine pack(am,index,job)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        dimension am(maxvec),index(maxdim)

C  If necessary, find and record the zero locations

        if (job.eq.0) then
          k=0
          do 10 j=1,6*nv
            if (am(indp(j,j)).ne.0.0.and.index(j).ne.0) then
              k=k+1
              index(j)=k
            else
              index(j)=0
            endif
10        continue
          rank=k
        endif

C  Now go through and remove the zeroes

        m=0
        n=0
        do 20 j=1,6*nv
        do 20 k=1,j
        m=m+1
          if (index(j).ne.0.and.index(k).ne.0) then
            n=n+1
            am(n)=am(m)
          endif
20      continue
        return
        end

C-------------------------------------------------------------
C
C  Function IDOF returns the location in the index array of
C    the D.O.F. associated with mode ix, iy of variable iv.

        integer function idof(ix,iy,iv)
```

```
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        idof=int((iv-1)*nv+(iy-1)*nm(1)+ix)
        return
        end
```

```
C  File: SPADBF.FOR
C-----------------------------------------------------------
C
C  Subroutine IBFUNC calculates the required Beam Function
C  integrals.  Originally written by Wilson Tsang.

C  Codes for Boundary Conditions are:

C       1       Simply Supported-Simply Supported
C       2       Clamped-Free
C       3       Clamped-Clamped
C       4       Free-Free (Includes Free Body Modes)
C       5       Simply Supported-Clamped
C       6       Simply Supported-Free (Does not include FBM)

        subroutine ibfunc(bcx,bcy)
        implicit real*8 (a-h,o-z)
        integer bcx,bcy,p,q
$include:'spad.inc'

        call bcons(1,bcx)
        if (bcx.eq.4) then
          call intgl1(1,psi,maxnx)
          else
          call intgl(1,psi,maxnx)
        endif
        call bcons(2,bcy)
        if (bcy.eq.4) then
          call intgl1(2,phi,maxny)
          else
          call intgl(2,phi,maxny)
        endif
        end


C-----------------------------------------------------------
C  Subroutine BCONS evaluates the coefficients for use in the
C  generalized beam functions (Dugundji).  Written by
C  Wilson Tsang.

        subroutine bcons(ixy,n)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'

        pi=3.14159265358979323846
        i=nm(ixy)
C  SS-SS
        if (n.eq.1) then
          do 10 j=1,i
          bcbeta(j,ixy)=((j+iodd*(j-1)))*pi
          bcb(j,ixy)=0.0
10        continue
          bcthet(ixy)=0.0
          bca(ixy)=0.0
```

```
          return
        endif

C  CL-FR
        if (n.eq.2) then
          do 20 j=1,i
          bcbeta(j,ixy)=((j+iodd*(j-1))-0.5)*pi
          bcb(j,ixy)=2.0*mod((j+iodd*(j-1)),2)-1
20        continue
          bcthet(ixy)=-pi/4.0
          bca(ixy)=1.0
          return
        endif

C  CL-CL
        if (n.eq.3) then
          do 30 j=1,i
          bcbeta(j,ixy)=((j+iodd*(j-1))+.5)*pi
          bcb(j,ixy)=2.0*mod((j+iodd*(j-1)),2)-1
30        continue
          bcthet(ixy)=-pi/4.0
          bca(ixy)=1.0
          return
        endif

C  FR-FR
        if (n.eq.4) then
          bcbeta(1,ixy)=0.0
          bcbeta(2,ixy)=0.0
          bcb(1,ixy)=0.0
          bcb(2,ixy)=0.0
          do 40 j=3-iodd,i
          bcbeta(j,ixy)=((j+iodd*(j-1))-1.5)*pi
          bcb(j,ixy)=2.0*mod((j+iodd*(j-1)),2)-1
40        continue
          bcthet(ixy)=3.0*pi/4
          bca(ixy)=1.0
          return
        endif

C  SS-CL
        if (n.eq.5) then
          do 50 j=1,i
          bcbeta(j,ixy)=((j+iodd*(j-1))+.25)*pi
          bcb(j,ixy)=2.0*mod((j+iodd*(j-1)),2)-1
50        continue
          bcthet(ixy)=0.0
          bca(ixy)=0.0
          return
        endif

C  SS-FR
        if (n.eq.6) then
          do 60 j=1,i
```

```
            bcbeta(j,ixy)=((j+iodd*(j-1))+.25)*pi
            bcb(j,ixy)=2.0*mod((j+iodd*(j-1))+1,2)-1
60          continue
            bcthet(ixy)=0.0
            bca(ixy)=0.0
            return
          endif
          end

C-------------------------------------------------------------
C  Subroutine INTGL formulates the integrals of products of
C  beam function and their derivatives.
C  F(p,q,m,n) is the integral of
C     the pth derivative of phi(m) *
C     the qth derivative of phi(n)

          subroutine intgl (ixy,f,nfsize)
          implicit real*8 (a-h,o-z)
$include:'spad.inc'
          dimension f(0:2,0:2,nfsize,nfsize)
          pi=3.14159265358979323846
          do 10 i=0,2
          do 10 j=0,2
          do 10 m=1,nm(ixy)
          do 10 n=1,nm(ixy)
          if (i.gt.j) then
            f(i,j,m,n)=f(j,i,n,m)
            goto 10
          endif
          d1=bcbeta(m,ixy)-bcbeta(n,ixy)
          d2=bcbeta(m,ixy)+bcbeta(n,ixy)
          d3=bcbeta(m,ixy)*bcbeta(m,ixy)+
     &      bcbeta(n,ixy)*bcbeta(n,ixy)
          d4=2.0*mod(i+1,2)-1
          d5=2.0*mod(j+1,2)-1
          d6=bcthet(ixy)+i*pi/2.0
          d7=bcthet(ixy)+j*pi/2.0
          if (m.eq.n) then
            d8=dcos(d6-d7)
          else
            d8=(dsin(d1+d6-d7)-dsin(d6-d7))/d1
          endif
          d9=(dsin(d6+d7)-dsin(d2+d6+d7))/d2
          d10=d4*bca(ixy)*(bcbeta(n,ixy)*dcos(d7)+
     &       bcbeta(m,ixy)*dsin(d7))
          d11=bcb(m,ixy)*(bcbeta(m,ixy)*dsin(bcbeta(n,ixy)+d7)-
     &       bcbeta(n,ixy)*dcos(bcbeta(n,ixy)+d7))
          d12=d5*bca(ixy)*(bcbeta(m,ixy)*dcos(d6)+
     &       bcbeta(n,ixy)*dsin(d6))
          d13=bcb(n,ixy)*(bcbeta(n,ixy)*dsin(bcbeta(m,ixy)+d6)-
     &       bcbeta(m,ixy)*dcos(bcbeta(m,ixy)+d6))
          d14=(d4*d5*bca(ixy)*bca(ixy)+
     &       bcb(m,ixy)*bcb(n,ixy))/d2
          f(i,j,m,n)=(bcbeta(m,ixy)**i)*(bcbeta(n,ixy)**j)*
```

```
     &     (d8+d9+d14+dsqrt(2.0)*(d10+d11+d12+d13)/d3)
           if (abs(f(i,j,m,n)).lt.0.1) f(i,j,m,n)=0.0
10         continue
           return
           end

C-------------------------------------------------------------
C  Subroutine INTGL1 is necessary to take into account the
C  peculiarities of the FR-FR mode.

           subroutine intgl1 (ixy,f,nfsize)
           implicit real*8 (a-h,o-z)
$include:'spad.inc'
           dimension f(0:2,0:2,nfsize,nfsize)
           pi=3.14159265358979323846
           r2=dsqrt(2.0)
           r3=dsqrt(3.0)
           do 30 i=0,2
           do 30 j=0,2
           do 10 m=1,2-iodd
           do 10 n=1,nm(ixy)
           f(i,j,m,n)=0.0
           f(j,i,m,n)=0.0
10         continue
           do 20 m=1,nm(ixy)
           do 20 n=1,2-iodd
           f(i,j,m,n)=0.0
           f(j,i,m,n)=0.0
20         continue
           do 30 m=3-iodd,nm(ixy)
           do 30 n=3-iodd,nm(ixy)
           if (i.gt.j) then
             f(i,j,m,n)=f(j,i,n,m)
             goto 30
           endif
           d1=bcbeta(m,ixy)-bcbeta(n,ixy)
           d2=bcbeta(m,ixy)+bcbeta(n,ixy)
           d3=bcbeta(m,ixy)*bcbeta(m,ixy)+
     &       bcbeta(n,ixy)*bcbeta(n,ixy)
           d4=2.0*mod(i+1,2)-1
           d5=2.0*mod(j+1,2)-1
           d6=bcthet(ixy)+i*pi/2.0
           d7=bcthet(ixy)+j*pi/2.0
           if (m.eq.n) then
             d8=dcos(d6-d7)
           else
             d8=(dsin(d1+d6-d7)-dsin(d6-d7))/d1
           endif
           d9=(dsin(d6+d7)-dsin(d2+d6+d7))/d2
           d10=d4*bca(ixy)*(bcbeta(n,ixy)*dcos(d7)+
     &       bcbeta(m,ixy)*dsin(d7))
           d11=bcb(m,ixy)*(bcbeta(m,ixy)*dsin(bcbeta(n,ixy)+d7)-
     &       bcbeta(n,ixy)*dcos(bcbeta(n,ixy)+d7))
           d12=d5*bca(ixy)*(bcbeta(m,ixy)*dcos(d6)+
```

```
     &       bcbeta(n,ixy)*dsin(d6))
        d13=bcb(n,ixy)*(bcbeta(n,ixy)*dsin(bcbeta(m,ixy)+d6)-
     &       bcbeta(m,ixy)*dcos(bcbeta(m,ixy)+d6))
        d14=(d4*d5*bca(ixy)*bca(ixy)+
     &       bcb(m,ixy)*bcb(n,ixy))/d2
         f(i,j,m,n)=(bcbeta(m,ixy)**i)*(bcbeta(n,ixy)**j)*
     &      (d8+d9+d14+r2*(d10+d11+d12+d13)/d3)
        if (abs(f(i,j,m,n)).lt.0.1) f(i,j,m,n)=0.0
30      continue

        do 40 i=3-iodd,nm(ixy)
        d1=dsin(bcbeta(i,ixy)+bcthet(ixy))+dsin(bcthet(ixy))
        d2=dsin(bcbeta(i,ixy)+bcthet(ixy))-dsin(bcthet(ixy))
        d3=dcos(bcbeta(i,ixy)+bcthet(ixy))-dcos(bcthet(ixy))
        f(0,1,1,i)=r2*d2+bcb(i,ixy)-bca(ixy)
        f(1,0,i,1)=f(0,1,1,i)
        f(0,2,1,i)=bcbeta(i,ixy)*(r2*d3+bca(ixy)+bcb(i,ixy))
        f(2,0,i,1)=f(0,2,1,i)
        if (iodd.eq.0) then
          f(0,1,2,i)=r3*(-r2*d1-bcb(i,ixy)-
     &      bca(ixy)+2/bcbeta(i,ixy)*
     &      (-r2*d3+bca(ixy)+bcb(i,ixy)))
          f(1,0,i,2)=f(0,1,2,i)
          f(1,0,2,i)=2*r3/bcbeta(i,ixy)*
     &      (r2*d3-bca(ixy)-bcb(i,ixy))
          f(0,1,i,2)=f(1,0,2,i)
          f(0,2,2,i)=r3*f(0,2,1,i)-2*r3*(bcbeta(i,ixy)*
     &       (r2*dcos(bcbeta(i,ixy)+bcthet(ixy))+bcb(i,ixy))-
     &       r2*d2-bcb(i,ixy)+bca(ixy))
          f(2,0,i,2)=f(0,2,2,i)
          f(1,2,2,i)=-2*r3*bcbeta(i,ixy)*
     &       (r2*d3+bca(ixy)+bcb(i,ixy))
          f(2,1,i,2)=f(1,2,2,i)
          f(1,1,2,i)=2*r3*(-r2*d2-bcb(i,ixy)+bca(ixy))
          f(1,1,i,2)=f(1,1,2,i)
        endif
        do 40 j=0,2
        do 40 m=0,2
        do 40 n=1,2-iodd
        if (abs(f(j,m,n,i)).lt.0.1) f(j,m,n,i)=0.0
        if (abs(f(j,m,i,n)).lt.0.1) f(j,m,i,n)=0.0
40      continue
        f(0,0,1,1)=1.0
        if (iodd.eq.0) then
          f(0,0,2,2)=1.0
          f(0,1,1,2)=-2.0*r3
          f(1,0,2,1)=-2.0*r3
          f(1,1,2,2)=12.0
        endif
        return
        end
```

```
C  File:  SPADLP.FOR
C  This are LINPACK routines required by SPAD

      subroutine dppco(ap,n,rcond,z,info)
      integer n,info
      double precision ap(1),z(1)
      double precision rcond
c
c     dppco factors a double precision symmetric positive
c     definite matrix stored in packed form
c     and estimates the condition of the matrix.
c
c     if  rcond  is not needed, dppfa is slightly faster.
c     to solve  a*x = b , follow dppco by dppsl.
c     to compute  inverse(a)*c , follow dppco by dppsl.
c     to compute  determinant(a) , follow dppco by dppdi.
c     to compute  inverse(a) , follow dppco by dppdi.
c
c     on entry
c
c        ap      double precision (n*(n+1)/2)
c                the packed form of a symmetric matrix  a.
c                the columns of the upper triangle are stored
c                sequentially in a one-dimensional array of
c                length  n*(n+1)/2 .
c                see comments below for details.
c
c        n       integer
c                the order of the matrix  a .
c
c     on return
c
c        ap      an upper triangular matrix  r , stored in
c                packed form, so that  a = trans(r)*r .
c                if  info .ne. 0 , the factorization
c                is not complete.
c
c        rcond   double precision
c                an estimate of the reciprocal condition of
a .
c                for the system  a*x = b , relative
c                perturbations in  a  and  b  of size
c                epsilon  may cause relative perturbations
c                in  x  of size  epsilon/rcond .
c                if  rcond  is so small that the logical
c                expression
c                          1.0 + rcond .eq. 1.0
c                is true, then  a  may be singular to working
c                precision.  in particular,  rcond  is zero
c                if exact singularity is detected or the
c                estimate underflows.  if info .ne. 0 ,
c                rcond is unchanged.
c
c        z       double precision(n)
```

```
c                a work vector whose contents are usually
c                unimportant.
c                if  a  is singular to working precision,
c                then  z  is an approximate null vector in
c                the sense that
c                    norm(a*z) = rcond*norm(a)*norm(z) .
c                if  info .ne. 0 , z  is unchanged.
c
c        info    integer
c                = 0  for normal return.
c                = k  signals an error condition.  the
c                     leading minor of order  k  is not
c                     positive definite.
c
c     packed storage
c
c          the following program segment will pack the upper
c          triangle of a symmetric matrix.
c
c                k = 0
c                do 20 j = 1, n
c                   do 10 i = 1, j
c                      k = k + 1
c                      ap(k) = a(i,j)
c             10    continue
c             20 continue
c
c     linpack.  this version dated 08/14/78 .
c     cleve moler, university of new mexico,
c       argonne national lab.
c
c     subroutines and functions
c
c     linpack dppfa
c     blas daxpy,ddot,dscal,dasum
c     fortran dabs,dmax1,dreal,dsign
c
c     internal variables
c
      double precision ddot,ek,t,wk,wkm
      double precision anorm,s,dasum,sm,ynorm
      integer i,ij,j,jm1,j1,k,kb,kj,kk,kp1
c
c
c     find norm of a
c
      j1 = 1
      do 30 j = 1, n
         z(j) = dasum(j,ap(j1),1)
         ij = j1
         j1 = j1 + j
         jm1 = j - 1
         if (jm1 .lt. 1) go to 20
         do 10 i = 1, jm1
```

```
            z(i) = z(i) + dabs(ap(ij))
            ij = ij + 1
   10    continue
   20    continue
   30 continue
      anorm = 0.0d0
      do 40 j = 1, n
         anorm = dmax1(anorm,z(j))
   40 continue
c
c     factor
c
      call dppfa(ap,n,info)
      if (info .ne. 0) go to 180
c
c        rcond = 1/(norm(a)*(estimate of norm(inverse(a)))) .
c        estimate = norm(z)/norm(y)
c          where  a*z = y  and  a*y = e .
c        the components of  e  are chosen to cause maximum
c        local growth in the elements of w
c        where  trans(r)*w = e .
c        the vectors are frequently rescaled to avoid
c        overflow.
c
c        solve trans(r)*w = e
c
         ek = 1.0d0
         do 50 j = 1, n
            z(j) = 0.0d0
   50    continue
         kk = 0
         do 110 k = 1, n
            kk = kk + k
            if (z(k) .ne. 0.0d0) ek = dsign(ek,-z(k))
            if (dabs(ek-z(k)) .le. ap(kk)) go to 60
               s = ap(kk)/dabs(ek-z(k))
               call dscal(n,s,z,1)
               ek = s*ek
   60       continue
            wk = ek - z(k)
            wkm = -ek - z(k)
            s = dabs(wk)
            sm = dabs(wkm)
            wk = wk/ap(kk)
            wkm = wkm/ap(kk)
            kp1 = k + 1
            kj = kk + k
            if (kp1 .gt. n) go to 100
               do 70 j = kp1, n
                  sm = sm + dabs(z(j)+wkm*ap(kj))
                  z(j) = z(j) + wk*ap(kj)
                  s = s + dabs(z(j))
                  kj = kj + j
   70          continue
```

```
                  if (s .ge. sm) go to 90
                     t = wkm - wk
                     wk = wkm
                     kj = kk + k
                     do 80 j = kp1, n
                        z(j) = z(j) + t*ap(kj)
                        kj = kj + j
   80                continue
   90             continue
  100          continue
               z(k) = wk
  110       continue
            s = 1.0d0/dasum(n,z,1)
            call dscal(n,s,z,1)
c
c           solve r*y = w
c
            do 130 kb = 1, n
               k = n + 1 - kb
               if (dabs(z(k)) .le. ap(kk)) go to 120
                  s = ap(kk)/dabs(z(k))
                  call dscal(n,s,z,1)
  120          continue
               z(k) = z(k)/ap(kk)
               kk = kk - k
               t = -z(k)
               call daxpy(k-1,t,ap(kk+1),1,z(1),1)
  130       continue
            s = 1.0d0/dasum(n,z,1)
            call dscal(n,s,z,1)
c
            ynorm = 1.0d0
c
c           solve trans(r)*v = y
c
            do 150 k = 1, n
               z(k) = z(k) - ddot(k-1,ap(kk+1),1,z(1),1)
               kk = kk + k
               if (dabs(z(k)) .le. ap(kk)) go to 140
                  s = ap(kk)/dabs(z(k))
                  call dscal(n,s,z,1)
                  ynorm = s*ynorm
  140          continue
               z(k) = z(k)/ap(kk)
  150       continue
            s = 1.0d0/dasum(n,z,1)
            call dscal(n,s,z,1)
            ynorm = s*ynorm
c
c           solve r*z = v
c
            do 170 kb = 1, n
               k = n + 1 - kb
               if (dabs(z(k)) .le. ap(kk)) go to 160
```

```
               s = ap(kk)/dabs(z(k))
               call dscal(n,s,z,1)
               ynorm = s*ynorm
  160       continue
            z(k) = z(k)/ap(kk)
            kk = kk - k
            t = -z(k)
            call daxpy(k-1,t,ap(kk+1),1,z(1),1)
  170    continue
c        make znorm = 1.0
         s = 1.0d0/dasum(n,z,1)
         call dscal(n,s,z,1)
         ynorm = s*ynorm
c
         if (anorm .ne. 0.0d0) rcond = ynorm/anorm
         if (anorm .eq. 0.0d0) rcond = 0.0d0
  180 continue
      return
      end
      subroutine dppfa(ap,n,info)
      integer n,info
      double precision ap(1)
c
c     dppfa factors a double precision symmetric positive
c     definite matrix stored in packed form.
c
c     dppfa is usually called by dppco, but it can be called
c     directly with a saving in time if  rcond  is not
c     needed.
c     (time for dppco) = (1 + 18/n)*(time for dppfa) .
c
c     on entry
c
c        ap      double precision (n*(n+1)/2)
c                the packed form of a symmetric matrix  a .
c                the columns of the upper triangle are
c                stored sequentially in a one-dimensional
c                array of length  n*(n+1)/2 .
c                see comments below for details.
c
c        n       integer
c                the order of the matrix  a .
c
c     on return
c
c        ap      an upper triangular matrix  r , stored in
c                packed form, so that  a = trans(r)*r .
c
c        info    integer
c                = 0  for normal return.
c                = k  if the leading minor of order  k  is
c                     not positive definite.
c
c
```

```
c     packed storage
c
c          the following program segment will pack the upper
c          triangle of a symmetric matrix.
c
c                k = 0
c                do 20 j = 1, n
c                   do 10 i = 1, j
c                      k = k + 1
c                      ap(k) = a(i,j)
c             10    continue
c             20 continue
c
c     linpack.  this version dated 08/14/78 .
c     cleve moler, university of new mexico,
c       argonne national lab.
c
c     subroutines and functions
c
c     blas ddot
c     fortran dsqrt
c
c     internal variables
c
      double precision ddot,t
      double precision s
      integer j,jj,jm1,k,kj,kk
c     begin block with ...exits to 40
c
c
         jj = 0
         do 30 j = 1, n
            info = j
            s = 0.0d0
            jm1 = j - 1
            kj = jj
            kk = 0
            if (jm1 .lt. 1) go to 20
            do 10 k = 1, jm1
               kj = kj + 1
               t = ap(kj) - ddot(k-1,ap(kk+1),1,ap(jj+1),1)
               kk = kk + k
               t = t/ap(kk)
               ap(kj) = t
               s = s + t*t
   10       continue
   20       continue
            jj = jj + j
            s = ap(jj) - s
c     ......exit
            if (s .le. 0.0d0) go to 40
            ap(jj) = dsqrt(s)
   30    continue
         info = 0
```

```
   40 continue
      return
      end

      subroutine dppdi(ap,n,det,job)
      integer n,job
      double precision ap(1)
      double precision det(2)
c
c     dppdi computes the determinant and inverse
c     of a double precision symmetric positive definite
c     matrix using the factors computed by dppco or dppfa .
c
c     on entry
c
c        ap      double precision (n*(n+1)/2)
c                the output from dppco or dppfa.
c
c        n       integer
c                the order of the matrix  a .
c
c        job     integer
c                = 11   both determinant and inverse.
c                = 01   inverse only.
c                = 10   determinant only.
c
c     on return
c
c        ap      the upper triangular half of the inverse .
c                the strict lower triangle is unaltered.
c
c        det     double precision(2)
c                determinant of original matrix if requested.
c                otherwise not referenced.
c                determinant = det(1) * 10.0**det(2)
c                with  1.0 .le. det(1) .lt. 10.0
c                or  det(1) .eq. 0.0 .
c
c     error condition
c
c        a division by zero will occur if the input factor
c        contains a zero on the diagonal and the inverse is
c        requested.
c        it will not occur if the subroutines are called
c        correctly and if dpoco or dpofa has set info .eq. 0
c
c     linpack.  this version dated 08/14/78 .
c     cleve moler, university of new mexico,
c       argonne national lab.
c
c     subroutines and functions
c
c     blas daxpy,dscal
c     fortran mod
```

```
c
c     internal variables
c
      double precision t
      double precision s
      integer i,ii,j,jj,jm1,j1,k,kj,kk,kp1,k1
c
c     compute determinant
c
      if (job/10 .eq. 0) go to 70
         det(1) = 1.0d0
         det(2) = 0.0d0
         s = 10.0d0
         ii = 0
         do 50 i = 1, n
            ii = ii + i
            det(1) = ap(ii)**2*det(1)
c           ...exit
            if (det(1) .eq. 0.0d0) go to 60
   10       if (det(1) .ge. 1.0d0) go to 20
               det(1) = s*det(1)
               det(2) = det(2) - 1.0d0
            go to 10
   20       continue
   30       if (det(1) .lt. s) go to 40
               det(1) = det(1)/s
               det(2) = det(2) + 1.0d0
            go to 30
   40       continue
   50    continue
   60    continue
   70 continue
c
c     compute inverse(r)
c
      if (mod(job,10) .eq. 0) go to 140
         kk = 0
         do 100 k = 1, n
            k1 = kk + 1
            kk = kk + k
            ap(kk) = 1.0d0/ap(kk)
            t = -ap(kk)
            call dscal(k-1,t,ap(k1),1)
            kp1 = k + 1
            j1 = kk + 1
            kj = kk + k
            if (n .lt. kp1) go to 90
            do 80 j = kp1, n
               t = ap(kj)
               ap(kj) = 0.0d0
               call daxpy(k,t,ap(k1),1,ap(j1),1)
               j1 = j1 + j
               kj = kj + j
   80       continue
```

```
   90       continue
  100    continue
c
c        form  inverse(r) * trans(inverse(r))
c
         jj = 0
         do 130 j = 1, n
            j1 = jj + 1
            jj = jj + j
            jm1 = j - 1
            k1 = 1
            kj = j1
            if (jm1 .lt. 1) go to 120
            do 110 k = 1, jm1
               t = ap(kj)
               call daxpy(k,t,ap(j1),1,ap(k1),1)
               k1 = k1 + k
               kj = kj + 1
  110       continue
  120       continue
            t = ap(jj)
            call dscal(j,t,ap(j1),1)
  130    continue
  140 continue
      return
      end

      subroutine daxpy(n,da,dx,incx,dy,incy)
c
c     constant times a vector plus a vector.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
      double precision dx(1),dy(1),da
      integer i,incx,incy,ixiy,m,mp1,n
c
      if(n.le.0)return
      if (da .eq. 0.0d0) return
      if(incx.eq.1.and.incy.eq.1)go to 20
c
c        code for unequal increments or equal increments
c          not equal to 1
c
      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        dy(iy) = dy(iy) + da*dx(ix)
        ix = ix + incx
        iy = iy + incy
   10 continue
      return
c
```

```
c        code for both increments equal to 1
c
c
c        clean-up loop
c
   20 m = mod(n,4)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dy(i) = dy(i) + da*dx(i)
   30 continue
      if( n .lt. 4 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,4
        dy(i) = dy(i) + da*dx(i)
        dy(i + 1) = dy(i + 1) + da*dx(i + 1)
        dy(i + 2) = dy(i + 2) + da*dx(i + 2)
        dy(i + 3) = dy(i + 3) + da*dx(i + 3)
   50 continue
      return
      end

      double precision function ddot(n,dx,incx,dy,incy)
c
c     forms the dot product of two vectors.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
      double precision dx(1),dy(1),dtemp
      integer i,incx,incy,ix,iy,m,mp1,n
c
      ddot = 0.0d0
      dtemp = 0.0d0
      if(n.le.0)return
      if(incx.eq.1.and.incy.eq.1)go to 20
c
c        code for unequal increments or equal increments
c          not equal to 1
c
      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        dtemp = dtemp + dx(ix)*dy(iy)
        ix = ix + incx
        iy = iy + incy
   10 continue
      ddot = dtemp
      return
c
c        code for both increments equal to 1
c
c
c        clean-up loop
```

```
c
   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dtemp = dtemp + dx(i)*dy(i)
   30 continue
      if( n .lt. 5 ) go to 60
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        dtemp = dtemp + dx(i)*dy(i) + dx(i + 1)*dy(i + 1) +
     *   dx(i + 2)*dy(i + 2) + dx(i + 3)*dy(i + 3) +
     *   dx(i + 4)*dy(i + 4)
   50 continue
   60 ddot = dtemp
      return
      end

      subroutine  dscal(n,da,dx,incx)
c
c     scales a vector by a constant.
c     uses unrolled loops for increment equal to one.
c     jack dongarra, linpack, 3/11/78.
c
      double precision da,dx(1)
      integer i,incx,m,mp1,n,nincx
c
      if(n.le.0)return
      if(incx.eq.1)go to 20
c
c        code for increment not equal to 1
c
      nincx = n*incx
      do 10 i = 1,nincx,incx
        dx(i) = da*dx(i)
   10 continue
      return
c
c        code for increment equal to 1
c
c
c        clean-up loop
c
   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dx(i) = da*dx(i)
   30 continue
      if( n .lt. 5 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        dx(i) = da*dx(i)
        dx(i + 1) = da*dx(i + 1)
        dx(i + 2) = da*dx(i + 2)
        dx(i + 3) = da*dx(i + 3)
```

```
        dx(i + 4) = da*dx(i + 4)
   50 continue
      return
      end

      double precision function dasum(n,dx,incx)
c
c     takes the sum of the absolute values.
c     jack dongarra, linpack, 3/11/78.
c
      double precision dx(1),dtemp
      integer i,incx,m,mp1,n,nincx
c
      dasum = 0.0d0
      dtemp = 0.0d0
      if(n.le.0)return
      if(incx.eq.1)go to 20
c
c        code for increment not equal to 1
c
      nincx = n*incx
      do 10 i = 1,nincx,incx
        dtemp = dtemp + dabs(dx(i))
   10 continue
      dasum = dtemp
      return
c
c        code for increment equal to 1
c
c
c        clean-up loop
c
   20 m = mod(n,6)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dtemp = dtemp + dabs(dx(i))
   30 continue
      if( n .lt. 6 ) go to 60
   40 mp1 = m + 1
      do 50 i = mp1,n,6
        dtemp = dtemp + dabs(dx(i)) + dabs(dx(i + 1)) +
     *  dabs(dx(i + 2)) + dabs(dx(i + 3)) +
     *  dabs(dx(i + 4)) + dabs(dx(i + 5))
   50 continue
   60 dasum = dtemp
      return
      end
```

```
C  File:  LISA.FOR
         program lisa
         implicit real*8 (a-h,o-z)
$include:'spad.inc'
         real*8 intgl,lsh
         logical success
         character*4 ext1,ext2
         dimension q(maxdim),modes(3,maxdim)
         dimension e(0:50,0:50),e11(0:50,0:50),e22(0:50,0:50)
         dimension e12(0:50,0:50)
         common /local/ alpha,intgl(0:2,0:2,2,0:2,0:2,2),
     &          supint(5),bm,cm
         common /local2/ bintgl(6,2,6,2,0:8),bsup(5,0:8)
         external streng,strain
         write (*,10)
10       format (/'  -- LOCAL INDENTATION STRAIN ANALYSIS --')

         write (*,*) 'Reading input files . . .'
         call input
         call getq(q,alpha,modes)
         call getint(bintgl,bsup)

C  Solve for the local indentation

         x1=5.0
         x2=4.0
         ftol=1d-10
         call mnbrak(x1,x2,x3,f1,f2,f3,streng)
         fret=brent(x1,x2,x3,streng,ftol,xmin)

         sp=xmin
         write (*,20) sp,fret
20       format (' Shape Parameter: ',g16.8/
     &           ' Strain Energy: ',g16.8)

C  Now we've got the answer, plot the local shape

         call filext(fname,'.lsh')
         open (unit=3,file=fname)

         do 100 ix=-50,50
           x=ix/50.0d0
           y1=alpha*w(x,sp)
           write (3,*) x,y1
100      continue
         close (unit=3)

C  And save the local modal amplitudes

         call filext(fname,'.lq ')
         open (unit=3,file=fname)
         write (3,*) sp,alpha,bm,cm
         close (unit=3)
```

```
9000    end

C-------------------------------------------------------------
C
C  Function W returns the shape of the impact mode at the
C  point x with shape parameter sp.

        real*8 function w(x,sp)
        implicit real*8 (a-h,o-z)
        t=sp*abs(x)
        w=exp(-t)*(sin(t)+cos(t))
        return
        end

C-------------------------------------------------------------
C
C  Function STRENG returns the strain energy given a value
C  of the shape parameter x.

        real*8 function streng(x)
        implicit real*8 (a-h,o-z)
        real*8 intgl
        integer*1 lookup
        dimension sp(0:8)
        dimension lookup(0:2,0:2)
        dimension p(3,2),y(3),q(2)
        common /local/ alpha,intgl(0:2,0:2,2,0:2,0:2,2),
     &         supint(5),bm,cm
        common /local2/ bintgl(6,2,6,2,0:8),bsup(5,0:8)
        external func
        data lookup /1,2,4,3,6,0,5,0,0/

C  Calculate powers of the shape parameter

        sp(0)=1.0d0/(x*x)
        do 100 i=1,8
          sp(i)=sp(i-1)*x
100     continue

C  Generate the new intgl values for this value of the
C  shape parameter

        do 120 nx=0,2
        do 120 ny=0,2
        do 120 ip=1,2
        do 120 mx=0,2
        do 120 my=0,2
        do 120 jp=1,2
          if (nx+ny.le.2.and.mx+my.le.2) then
            t=0.0
            do 110 k=0,8
              t=t+bintgl(lookup(nx,ny),ip,
     &           lookup(mx,my),jp,k)*sp(k)
110         continue
```

```
            intgl(nx,ny,ip,mx,my,jp)=t
          endif
120     continue

C  Don't forget supint values as well

        do 140 i=1,5
          t=0.0
          do 130 k=0,8
            t=t+bsup(i,k)*sp(k)
130       continue
          supint(i)=t
140     continue

C  Minimize over the u and v variables (given sp).

        t=1.0d-4
        p(1,1)=t
        p(1,2)=0.0d0
        p(2,1)=0.0d0
        p(2,2)=t
        p(3,1)=t
        p(3,2)=t
        do 150 i=1,3
          q(1)=p(i,1)
          q(2)=p(i,2)
          y(i)=func(q)
150     continue
        ftol=1e-6
        call amoeba(p,y,3,2,2,ftol,func,iter)

        ilo=1
        do 160 i=2,3
          if (y(i).lt.y(ilo)) ilo=i
160     continue
        write (*,*) y(ilo)
        streng=y(ilo)
        bm=p(ilo,1)
        cm=p(ilo,2)
        return
        end

C-------------------------------------------------------------
C
C  Function FUNC is the total strain energy
C  This routine returns U as a function of bi=ci and
C  assumes that the integral tables have been updated
C  to the curren value of the shape parameter sp.

        real*8 function func(x)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        real*8 intgl
        dimension x(2)
```

```
      common /local/ alpha,intgl(0:2,0:2,2,0:2,0:2,2),
     &       supint(5),bm,cm

      z=za
      z2=z*z
      z3=z2*z
      z4=z3*z

      ai=alpha
      bi=x(1)
      ci=x(2)

      a2=ai*ai
      b2=bi*bi
      c2=ci*ci
      t=0.0d0

C  U1:  terms with A11
        t=t+aa(1)/2*(z2*bi*bi*intgl(2,0,1,2,0,1)
     &  + z3*bi*b2*intgl(2,0,1,2,0,2)
     &  + z3*bi*c2*intgl(2,0,1,1,1,2)
     &  + z*bi*a2*intgl(2,0,1,1,0,2)
     &  + z4/4*b2*b2*intgl(2,0,2,2,0,2)
     &  + z4/2*b2*c2*intgl(2,0,2,1,1,2)
     &  + z2/2*b2*a2*intgl(2,0,2,1,0,2)
     &  + z4/4*c2*c2*intgl(1,1,2,1,1,2)
     &  + z2/2*c2*a2*intgl(1,1,2,1,0,2)
     &  + a2*a2/4*intgl(1,0,2,1,0,2))

C  U2:  terms with A12
        t=t+aa(3)*(z2*bi*ci*intgl(2,0,1,0,2,1)
     &  + z3/2*bi*b2*intgl(2,0,1,1,1,2)
     &  + z3/2*bi*c2*intgl(2,0,1,0,2,2)
     &  + z/2*bi*a2*intgl(2,0,1,0,1,2)
     &  + z3/2*b2*ci*intgl(2,0,2,0,2,1)
     &  + z4/4*b2*b2*intgl(2,0,2,1,1,2)
     &  + z4/4*b2*c2*intgl(2,0,2,0,2,2)
     &  + z2/4*b2*a2*intgl(2,0,2,0,1,2)
     &  + z3/2*c2*ci*intgl(1,1,2,0,2,1)
     &  + z4/4*c2*b2*intgl(1,1,2,1,1,2)
     &  + z4/4*c2*c2*intgl(1,1,2,0,2,2)
     &  + z2/4*c2*a2*intgl(1,1,2,0,1,2)
     &  + z/2*a2*ci*intgl(1,0,2,0,2,1)
     &  + z2/4*a2*b2*intgl(1,0,2,1,1,2)
     &  + z2/4*a2*c2*intgl(1,0,2,0,2,2)
     &  + a2*a2/4*intgl(1,0,2,0,1,2))

C  U3:  terms with A22
        t=t+aa(2)/2*(z2*ci*ci*intgl(0,2,1,0,2,1)
     &  + z3*ci*b2*intgl(0,2,1,1,1,2)
     &  + z3*ci*c2*intgl(0,2,1,0,2,2)
     &  + z*ci*a2*intgl(0,2,1,0,1,2)
     &  + z4/4*b2*b2*intgl(1,1,2,1,1,2)
     &  + z4/2*b2*c2*intgl(1,1,2,0,2,2)
```

```
     &   + z2/2*b2*a2*intgl(1,1,2,0,1,2)
     &   + z4/4*c2*c2*intgl(0,2,2,0,2,2)
     &   + z2/2*c2*a2*intgl(0,2,2,0,1,2)
     &   + a2*a2/4*intgl(0,1,2,0,1,2))

C  U4, U5  are zero due to orthotropic nature of the problem

C  U6:  terms with A66

           t1=z2*bi*(bi+ci)*intgl(1,1,1,1,1,1)
     &   + 2*z2*bi*(b2+c2)*supint(1)
     &   + 2*z*bi*a2*supint(2)
     &   + z2*ci*ci*intgl(1,1,1,1,1,1)
     &   + 2*z3*ci*(b2+c2)*supint(1)
     &   + 2*z*ci*a2*supint(2)
           t1=t1+z4*b2*b2*supint(3)
     &   + 2*z4*b2*c2*supint(3)
     &   + 2*z2*b2*a2*supint(4)
     &   + z4*b2*b2*supint(3)
     &   + 2*z2*c2*a2*supint(4)
     &   + a2*a2*supint(5)
           t=t+aa(6)/2*t1

C  U7:  terms with Ezz
           t=t+Ezz/2/h*ai*ai*intgl(0,0,1,0,0,1)

C  U8:  terms with Gyz
           t=t+Gyz*h/2*(ai*ai*intgl(0,1,1,0,1,1)/3
     &   + ai*bi*intgl(0,1,1,1,0,1)*5/12
     &   + bi*bi*intgl(1,0,1,1,0,1)/2)

C  U9:  terms with Gxz
           t=t+Gyz*h/2*(ai*ai*intgl(1,0,1,1,0,1)/3
     &   + ai*ci*intgl(1,0,1,0,1,1)*5/12
     &   + ci*ci*intgl(0,1,1,0,1,1)/2)

           func=t
           return
           end

C-------------------------------------------------------------
C
C  Subroutine FILEXT appends the extension ext to name.
C
           subroutine filext(name,ext)
           character*40 name
           character*4 ext
           j=0
           do 10 i=1,36
             if (name(i:i).eq.'.'
     &           .or.name(i:i).eq.' '.and.j.eq.0) j=i
10         continue
           if (j.eq.0) then
             write (*,*) 'File name too long.'
```

```
          stop
        endif

        name(j:j+3)=ext(1:4)
        name(j+4:j+4)=' '
        return
        end

C-------------------------------------------------------------
C
C  Subroutine INPUT reads data from a prepared file for use
C  by the analysis programs SPAD and LISA.
C  These data files are created by PROBLEM.FOR and have
C  no extension.

        subroutine input
        implicit real*8 (a-h,o-z)
        integer bcx,bcy
$include:'spad.inc'
        character*40 line
        pi=3.14159265358979323846

        open(unit=2,file=' ',err=110)
        read (2,10) line
10      format (a40)
        j=1
20      j=j+1
        if (line(j:j).eq.' '.or.line(j:j).eq.09) goto 20
        k=j
30      k=k+1
        if (line(k:k).ne.' ') goto 30
        line(k:k)='.'
        fname(1:k-j+1)=line(j:k)
        read (2,*,err=100) a,b,bcx,bcy
        read (2,*,err=100) (aa(i),i=1,6)
        read (2,*,err=100) ta,rhoa
        read (2,*,err=100) (ab(i),i=1,6)
        read (2,*,err=100) tb,rhob
        read (2,*,err=100) Ezz,Gxz,Gyz,h,rhoc
        read (2,*,err=100) rhof,tf
        read (2,*,err=100) mi,v,hk,ri
        read (2,*,err=100) nm(1),nm(2)
        read (2,*,err=100) dt
        close (unit=2)
        zac=h/2
        zbc=-zac
        za=zac+ta/2
        zb=zbc-tb/2
        area=a*b
        di=sqrt(pi*ri*ri)
        al=0.5-di/2/a
        ah=0.5+di/2/a
        bl=0.5-di/2/b
        bh=0.5+di/2/b
```

```
        return

100     write (*,105) fname
105     format (' Error reading data from file: ',a40)
        goto 120
110     write (*,115) fname
115     format (' Error opening file: ',a40)
120     stop
        end

C-------------------------------------------------------------
C
C  Subroutine GETQ reads in the global plate displacements
C  and maximum approach generated by the SPAD program.

        subroutine getq(q,alpha,modes)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        dimension q(maxdim),modes(3,maxdim)
        call filext(fname,'.tq ')
        open (unit=2,file=fname,status='old')
        read (2,*) alpha,rank
        read (2,*) nm(1),nm(2)
        do 20 i=1,rank
          read (2,*) (modes(j,i),j=1,3),q(i)
20      continue
        do 30 i=1,2
          read (2,*) (bcbeta(j,i),j=1,nm(i))
          read (2,*) bcthet(i),bca(i)
          read (2,*) (bcb(j,i),j=1,nm(i))
30      continue
        close (unit=2)
        return
        end

C-------------------------------------------------------------
C
C  Subroutine GETINT reads the table of integrals from the
C  INTGL program.

        subroutine getint(intgl,supint)
        implicit real*8 (a-h,o-z)
        real*8 intgl
        integer pi,pj
        dimension intgl(6,2,6,2,0:8)
        dimension supint(5,0:8)

        open (unit=2,file='intgl.tbl',status='old')
        do 40 i=1,6
        do 40 pi=1,2
        do 40 j=1,i
        do 40 pj=1,2
          read (2,*) (intgl(i,pi,j,pj,n),n=0,8)
          do 30 k=0,8
```

```
            intgl(j,pj,i,pi,k)=intgl(i,pi,j,pj,k)
30          continue
40        continue

          do 50 i=1,5
            read (2,*) (supint(i,n),n=0,8)
50        continue

          close (unit=2)
          return
          end
```

```
C  File:  DIFA.FOR
         program DIFA
         implicit real*8 (a-h,o-z)
$include:'spad.inc'
         integer iarray(2)
         real*8 lsh
         logical success
         character*4 ext1,ext2
         dimension q(maxdim),modes(3,maxdim)
         dimension e(0:50,0:50),e11(0:50,0:50),e22(0:50,0:50)
         dimension e12(0:50,0:50)
         common /local/ alpha,sp,bm,cm

         write (*,10)
10       format (/'  -- Damage Induced Failure Analysis --')

         write (*,*) 'Reading input files . . .'
         fname='                              '
         call input
         call getq(q,alpha,modes)
         call getlq

         call filext(fname,'.rs ')
         open (unit=3,file=fname)
         call filext(fname,'.   ')
         write (*,15) fname
         write (3,15) fname
15       format (' Residual Strength Analysis for file: 'a10/)

C  Calculate the undamaged strength:

C  The effective modulus of the film with core is 1.20e6 psi

         Efa=1.20e6
         Ea=(aa(1)-aa(3)*aa(3)/aa(2))/ta
         t=(ta*ta/2.0d0+Efa/Ea*tf*(ta+tf/2.0d0))/
     &     (ta+Efa/Ea*tf)
         da=abs(ta/2.0d0-t)
         df=abs(ta+tf/2.0d0-t)
         EI=Ea*((ta*ta*ta+Efa/Ea*tf*tf*tf)/12.0d0
     &          +(ta*da*da+Efa/Ea*tf*df*df))*b
         ck=Ezz*b/h*2.0d0
         P0=4.0d0*sqrt(ck*EI)
         s0=P0/(ta+tb)/b
         write (*,20) P0,s0
         write (3,20) P0,s0
20       format (' Undamaged Strength:  ',f8.0,' lbs'/
     &           '                          ',f8.0,' psi'/)

C  Now figure out the strains:

         nxdiv=25
         nydiv=25
```

```
      write (*,*) 'Calculating Residual Strength. . .'
C  e11 in facesheet A:

      iarray(2)=25
      e11max=-1.0d10
      e11min=-e11max
      do 50 iy=0,nydiv
        y=xmap(iy)
        iarray(1)=iy
        do 40 it=1,2
          ix=iarray(it)
          x=xmap(ix)
          t1=0.0d0
          t2=0.0d0
          t3=0.0d0
          do 30 i=1,rank
            if (modes(3,i).eq.1) then
              t1=t1+q(i)*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),0,y)
            elseif (modes(3,i).eq.2) then
              t2=t2+q(i)*za*shape(1,modes(1,i),2,x)
     &             *shape(2,modes(2,i),0,y)
            elseif (modes(3,i).eq.3) then
              t3=t3+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
            endif
30        continue
          t1=t1/a+alpha*lsh(1,0,x,y,sp)
          t2=t2/(a*a)+bm*za*lsh(2,0,x,y,sp)
          t3=t3/(a*b)+cm*za*lsh(1,1,x,y,sp)
          t=t2+(t1*t1+t2*t2+t3*t3)/2.0d0
          e11(ix,iy)=t
          e11(50-ix,iy)=t
          e11(50-ix,50-iy)=t
          e11(ix,50-iy)=t
          e11min=min(e11min,t)
          e11max=max(e11max,t)
40      continue
50    continue

C  e22 in facesheet A:

      e22max=-1.0d10
      e22min=-e22max
      do 80 iy=0,nydiv
        y=xmap(iy)
        iarray(1)=iy
        do 70 it=1,2
          ix=iarray(it)
          x=xmap(ix)
          t1=0.0d0
          t2=0.0d0
          t3=0.0d0
          do 60 i=1,rank
```

```
            if (modes(3,i).eq.1) then
              t1=t1+q(i)*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.2) then
              t2=t2+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.3) then
              t3=t3+q(i)*za*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),2,y)
            endif
60        continue
          t1=t1/b+alpha*lsh(0,1,x,y,sp)
          t2=t2/(a*b)+bm*za*lsh(0,2,x,y,sp)
          t3=t3/(b*b)+cm*za*lsh(1,1,x,y,sp)
          t=t3+(t2*t2+t3*t3+t1*t1)/2.0d0
          e22(ix,iy)=t
          e22(50-ix,iy)=t
          e22(50-ix,50-iy)=t
          e22(ix,50-iy)=t
          e22min=min(e22min,t)
          e22max=max(e22max,t)
70      continue
80    continue

      e12max=0.0d10
      do 110 iy=0,nydiv
        y=xmap(iy)
        iarray(1)=iy
        do 100 it=1,2
          ix=iarray(it)
          x=xmap(ix)
          t1=0.0d0
          t2=0.0d0
          t3=0.0d0
          t4=0.0d0
          t5=0.0d0
          t6=0.0d0
          do 90 i=1,rank
            if (modes(3,i).eq.1) then
              t1=t1+q(i)*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),0,y)
              t2=t2+q(i)*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.2) then
              t3=t3+q(i)*za*shape(1,modes(1,i),2,x)
     &             *shape(2,modes(2,i),0,y)
              t4=t4+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.3) then
              t5=t5+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
              t6=t6+q(i)*za*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),2,y)
            endif
```

```
90          continue
            t1=t1/a+alpha*lsh(1,0,x,y,sp)
            t2=t2/b+alpha*lsh(0,1,x,y,sp)
            t3=t3/(a*a)+bm*za*lsh(2,0,x,y,sp)
            t4=t4/(a*b)+bm*za*lsh(1,1,x,y,sp)
            t5=t5/(a*b)+cm*za*lsh(1,1,x,y,sp)
            t6=t6/(b*b)+cm*za*lsh(0,2,x,y,sp)
            t=t4+t5+t3*t4+t5*t6+t1*t2
            e12(ix,iy)=t
            e12(50-ix,iy)=-t
            e12(50-ix,50-iy)=t
            e12(ix,50-iy)=-t
            e12max=max(e12max,abs(t))
100       continue
110     continue
        e12min=-e12max

        eult=.0125
        write (*,120)
120     foramt('For loads parallel to the fibers: (Mar-Lin)')
        write (3,120)

        i=25
        do 215 j=0,24
          if (e11(25,j).gt.e11(25,i)) i=j
215     continue

        if (e11(25,i).le.eult) then
          write (*,*) '  No damage predicted'
          write (3,*) '  No damage predicted'
        else
220       i=i-1
          if (i.le.1) then
            write (*,*) '  Catastrophic Failure predicted'
            write (3,*) '  Catastrophic Failure predicted'
            goto 250
          endif
          if (e11(25,i).gt.eult.or.e11(25,i-1).gt.eult) goto
220
          t=eult-e11(25,i)/(e11(25,i+1)-e11(25,i))
          t1=xmap(i)+t*(xmap(i+1)-xmap(i))
          dw=2.0d0*abs(0.50-t1)
          write (*,230) dw
          write (3,230) dw
230       format ('   Damage width: ',f6.4,' in')
          hc=S0/2.0d0
          d1=.2031
          d2=b/4.5714
          if (dw.le.d1) then
            scr=s0-1.0771*s0*dw
          elseif (dw.le.d2) then
            scr=hc*dw**(-0.28)
          else
            scr=0.14*s0*d2**(-1.28)*(b-dw)
```

```
          endif
          Pcr=scr*b*(ta+tb)
          write (*,240) Pcr,scr
          write (3,240) Pcr,scr
240       format ('   Residual Strength: ',f8.0,' lbs'/
     &    '                      ',f8.0,
     &    ' psi  [Net Section Stress]')
        endif

250     write (*,260)
        write (3,260)
260     format (/' For loads at 45 degrees to the fibers: ',
     &    '(Net-Section)')

        i=25
        e(i,i)=abs(e11(i,i)+e12(i,i)+e12(i,i)/2.0d0)
        do 265 j=0,24
          e(j,j)=abs(e11(j,j)+e12(j,j)+e12(j,j)/2.0d0)
          write (*,*) j,e(j,j)
          if (e(j,j).gt.e(i,i)) then
            i=j
            e(i,i)=abs(e11(i,i)+e12(i,i)+e12(i,i)/2.0d0)
          endif
265     continue

        if (e(i,i).le.eult) then
          write (*,*) '  No damage predicted'
          write (3,*) '  No damage predicted'
        else
270       i=i-1
          if (i.le.1) then
            write (*,*) '  Catastrophic Failure predicted'
            write (3,*) '  Catastrophic Failure predicted'
            goto 250
          endif
          e(i-1,i-1)=abs(e11(i-1,i-1)+e12(i-1,i-1)+
     &      e12(i-1,i-1)/2.0d0)
          e(i+1,i+1)=abs(e11(i+1,i+1)+e12(i+1,i+1)+
     &      e12(i+1,i+1)/2.0d0)
          e(i,i)=abs(e11(i,i)+e12(i,i)+e12(i,i)/2.0d0)
          if (e(i,i).gt.eult.or.e(i-1,i-1).gt.eult) goto 270
          t=eult-e(i,i)/(e(i+1,i+1)-e(i,i))
          t1=xmap(i)+t*(xmap(i+1)-xmap(i))
          dw=2.0d0*abs(0.50-t1)
          dw=dw/sqrt(2.0d0)
          write (*,230) dw
          write (3,230) dw
          Pcr=s0*(b-dw)*(ta+tb)
          write (*,240) Pcr,s0
          write (3,240) Pcr,s0
        endif
        write (*,*)
        write (3,*)
        close (unit=3)
```

```
C  e11 in facesheet A:

        write (*,*) 'Calculating e11 strains . . .'
        e11max=-1.0d10
        e11min=-e11max
        do 350 ix=0,nxdiv
          x=xmap(ix)
          do 340 iy=0,nydiv
            y=xmap(iy)
            t1=0.0d0
            t2=0.0d0
            t3=0.0d0
            do 330 i=1,rank
              if (modes(3,i).eq.1) then
                t1=t1+q(i)*shape(1,modes(1,i),1,x)
     &              *shape(2,modes(2,i),0,y)
              elseif (modes(3,i).eq.2) then
                t2=t2+q(i)*za*shape(1,modes(1,i),2,x)
     &              *shape(2,modes(2,i),0,y)
              elseif (modes(3,i).eq.3) then
                t3=t3+q(i)*za*shape(1,modes(1,i),1,x)
     &              *shape(2,modes(2,i),1,y)
              endif
330           continue
              t1=t1/a+alpha*lsh(1,0,x,y,sp)
              t2=t2/(a*a)+bm*za*lsh(2,0,x,y,sp)
              t3=t3/(a*b)+cm*za*lsh(1,1,x,y,sp)
              t=t2+(t1*t1+t2*t2+t3*t3)/2.0d0
              e11(ix,iy)=t
              e11(50-ix,iy)=t
              e11(50-ix,50-iy)=t
              e11(ix,50-iy)=t
              e11min=min(e11min,t)
              e11max=max(e11max,t)
340         continue
            write (*,345)
345         format ('.'\)
350       continue

C  e22 in facesheet A:

        write (*,*)
        write (*,*) 'Calculating e22 strains . . .'
        e22max=-1.0d10
        e22min=-e22max
        do 380 ix=0,nxdiv
          x=xmap(ix)
          do 370 iy=0,nydiv
            y=xmap(iy)
            t1=0.0d0
            t2=0.0d0
            t3=0.0d0
            do 360 i=1,rank
```

```
            if (modes(3,i).eq.1) then
              t1=t1+q(i)*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.2) then
              t2=t2+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.3) then
              t3=t3+q(i)*za*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),2,y)
            endif
360       continue
          t1=t1/b+alpha*lsh(0,1,x,y,sp)
          t2=t2/(a*b)+bm*za*lsh(0,2,x,y,sp)
          t3=t3/(b*b)+cm*za*lsh(1,1,x,y,sp)
          t=t3+(t2*t2+t3*t3+t1*t1)/2.0d0
          e22(ix,iy)=t
          e22(50-ix,iy)=t
          e22(50-ix,50-iy)=t
          e22(ix,50-iy)=t
          e22min=min(e22min,t)
          e22max=max(e22max,t)
370     continue
        write (*,345)
380   continue

      write (*,*)
      write (*,*) 'Calculating e12 strains . . .'
      e12max=0.0d10
      do 410 ix=0,nxdiv
        x=xmap(ix)
        do 400 iy=0,nydiv
          y=xmap(iy)
          t1=0.0d0
          t2=0.0d0
          t3=0.0d0
          t4=0.0d0
          t5=0.0d0
          t6=0.0d0
          do 390 i=1,rank
            if (modes(3,i).eq.1) then
              t1=t1+q(i)*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),0,y)
              t2=t2+q(i)*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.2) then
              t3=t3+q(i)*za*shape(1,modes(1,i),2,x)
     &             *shape(2,modes(2,i),0,y)
              t4=t4+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
            elseif (modes(3,i).eq.3) then
              t5=t5+q(i)*za*shape(1,modes(1,i),1,x)
     &             *shape(2,modes(2,i),1,y)
              t6=t6+q(i)*za*shape(1,modes(1,i),0,x)
     &             *shape(2,modes(2,i),2,y)
```

```
              endif
390           continue
              t1=t1/a+alpha*lsh(1,0,x,y,sp)
              t2=t2/b+alpha*lsh(0,1,x,y,sp)
              t3=t3/(a*a)+bm*za*lsh(2,0,x,y,sp)
              t4=t4/(a*b)+bm*za*lsh(1,1,x,y,sp)
              t5=t5/(a*b)+cm*za*lsh(1,1,x,y,sp)
              t6=t6/(b*b)+cm*za*lsh(0,2,x,y,sp)
              t=t4+t5+t3*t4+t5*t6+t1*t2
              e12(ix,iy)=t
              e12(50-ix,iy)=-t
              e12(50-ix,50-iy)=t
              e12(ix,50-iy)=-t
              e12max=max(e12max,abs(t))
400         continue
            write (*,345)
410       continue
          write (*,*)
          e12min=-e12max


          theta=0.0d0
          pi=4.0d0*atan(1.0d0)
500       write (*,510)
          oldth=theta
510       format (' Enter Angle Strain Output',
     &      ' [''end'' to end]: '\)
          read (*,520,err=9000) theta
520       format (f20.0)
          write (*,530)
530       format (' Enter 3 Letter File Extension: '\)
          read (*,540) ext1
540       format (a3)
          ext2(2:4)=ext1
          ext2(1:1)='.'

          call filext(fname,ext2)
          open(unit=3,file=fname)

          if (theta.eq.0.0) then
            call conplt(e11min,e11max,e11)
          elseif (theta.eq.90.0) then
            call conplt(e22min,e22max,e22)
          elseif (theta.eq.180.0) then
            call conplt(e12min,e12max,e12)
          else
            if (theta.ne.oldth) then
              emin=1.0d10
              emax=-emin
              c=cos(2.0d0*theta/180.0d0*pi)
              s=sin(2.0d0*theta/180.0d0*pi)
              do 560 ix=0,50
                do 550 iy=0,50
                  t=(e11(ix,iy)+e22(ix,iy)
```

```
     &            +(e11(ix,iy)-e22(ix,iy))*c
     &             +e12(ix,iy)*s)/2.0d0
                 e(ix,iy)=t
                 emax=max(t,emax)
                 emin=min(t,emin)
550            continue
560          continue
           endif
           call conplt(emin,emax,e)
         endif
         close(unit=3)
         goto 500

9000     end

C-------------------------------------------------------------
C
C  Function XMAP returns the mapping point for the
C  evaluation of stresses

         real*8 function xmap(i)
         implicit real*8 (a-h,o-z)
         t=(25-i)/25.0d0
         xmap=0.5d0*(1.0d0-t*t)
         return
         end

C-------------------------------------------------------------
C
C  Subroutine FILEXT appends the extension ext to name.
C
         subroutine filext(name,ext)
         character*40 name
         character*4 ext
         j=0
         do 10 i=1,36
           if (name(i:i).eq.'.'
     &          .or.name(i:i).eq.' '.and.j.eq.0) j=i
10       continue
         if (j.eq.0) then
           write (*,*) 'File name too long.'
           stop
         endif

         name(j:j+3)=ext(1:4)
         name(j+4:j+4)=' '
         return
         end

C-------------------------------------------------------------
C
C  Subroutine INPUT reads data from a prepared file for use
C  by the analysis programs SPAD and LISA.
C  These data files are created by PROBLEM.FOR and have
```

```
C  no extension.

        subroutine input
        implicit real*8 (a-h,o-z)
        integer bcx,bcy
$include:'spad.inc'
        character*40 line
        pi=3.14159265358979323846

        open(unit=2,file=' ',err=110)
        read (2,10) line
10      format (a40)
        j=1
20      j=j+1
        if (line(j:j).eq.' '.or.line(j:j).eq.09) goto 20
        k=j
30      k=k+1
        if (line(k:k).ne.' ') goto 30
        line(k:k)='.'
        fname(1:k-j+1)=line(j:k)
        read (2,*,err=100) a,b,bcx,bcy
        read (2,*,err=100) (aa(i),i=1,6)
        read (2,*,err=100) ta,rhoa
        read (2,*,err=100) (ab(i),i=1,6)
        read (2,*,err=100) tb,rhob
        read (2,*,err=100) Ezz,Gxz,Gyz,h,rhoc
        read (2,*,err=100) rhof,tf
        read (2,*,err=100) mi,v,hk,ri
        read (2,*,err=100) nm(1),nm(2)
        read (2,*,err=100) dt
        close (unit=2)
        zac=h/2
        zbc=-zac
        za=zac+ta/2
        zb=zbc-tb/2
        area=a*b
        di=sqrt(pi*ri*ri)
        al=0.5-di/2/a
        ah=0.5+di/2/a
        bl=0.5-di/2/b
        bh=0.5+di/2/b
        return

100     write (*,105) fname
105     format (' Error reading data from file: ',a40)
        goto 120
110     write (*,115) fname
115     format (' Error opening file: ',a40)
120     stop
        end

C-------------------------------------------------------------
C
C  Subroutine GETQ reads in the global plate displacements
```

```
C  and maximum approach generated by the SPAD program.

        subroutine getq(q,alpha,modes)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        dimension q(maxdim),modes(3,maxdim)
        call filext(fname,'.tq ')
        open (unit=2,file=fname,status='old')
        read (2,*) alpha,rank
        read (2,*) nm(1),nm(2)
        do 20 i=1,rank
          read (2,*) (modes(j,i),j=1,3),q(i)
20      continue
        do 30 i=1,2
          read (2,*) (bcbeta(j,i),j=1,nm(i))
          read (2,*) bcthet(i),bca(i)
          read (2,*) (bcb(j,i),j=1,nm(i))
30      continue
        close (unit=2)
        return
        end

C-----------------------------------------------------------
C
C  Subroutine GETLQ reads the local model modal amplitudes
C
        subroutine getlq
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        common /local/ alpha,sp,bm,cm
        call filext(fname,'.lq ')
        open (unit=2,file=fname)
        read (2,*) sp,alpha,bm,cm
        close (unit=2)
        return
        end

C-----------------------------------------------------------
C
C  Function SHAPE returns the value of:
C       the pth derivative of beam function n
C       in the i direction at point xi (normalized).

        real*8 function shape(i,n,p,x)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer i,n,p

        r2=sqrt(2.0)
        beta=bcbeta(n,i)
        if (beta.eq.0) then
          if (p.eq.0) then
            shape=1.0
          else
```

```
            shape=0.0
          endif
        else
          theta=beta*x+bcthet(i)
          e1=bca(i)*exp(-beta*x)
          e2=bcb(n,i)*exp(-beta*(1.0-x))
          if (p.eq.0) then
            shape=r2*dsin(theta)+e1+e2
          elseif (p.eq.1) then
            shape=beta*(r2*dcos(theta)-e1+e2)
          else
            shape=beta*beta*(-r2*dsin(theta)+e1+e2)
          endif
        endif
        return
        end

C-------------------------------------------------------------
C
C  Function LSH returns the local impact mode shape function.
C    The value returned is the ixth x derivative times
C    the iyth y derivative of the shape function (with shape
C    parameter sp) at the normalized point (xi,eta).
C    These coordinates are normalized [0,1] over the plate.

        real*8 function lsh(ix,iy,xi,eta,sp)
        implicit real*8 (a-h,o-z)
$include:'spad.inc'
        integer*1 ak(7,2)
        dimension t(2)

        x=a*(xi-0.5d0)
        y=b*(eta-0.5d0)
        r=sqrt(x*x+y*y)
        z=sp*r
        call defshp(ix,iy,ak(1,1),ak(1,2))
        if (r.ne.0.0) then
          theta=atan2(y,x)
          do 10 i=1,2
            t(i)=ak(1,i)
            if (ak(2,i).eq.1) then
              t(i)=t(i)*exp(-z)*(cos(z)+sin(z))
            elseif (ak(3,i).eq.1) then
              t(i)=-t(i)*2.0d0*sp*exp(-z)*sin(z)
            elseif (ak(4,i).eq.1) then
              t(i)=t(i)*2.0d0*sp*sp*exp(-z)*(sin(z)-cos(z))
            endif
            t(i)=t(i)*sin(theta)**ak(5,i)
     &           *cos(theta)**ak(6,i)*r**ak(7,i)
10        continue
          lsh=t(1)+t(2)
        else
          do 20 i=1,2
            t(i)=ak(1,i)
```

```
            if (ak(3,i).eq.1.and.ak(7,i).eq.0) then
              t(i)=0.0d0
            elseif (ak(3,i).eq.1.and.ak(7,i).eq.-1) then
              t(i)=-t(i)*2.0d0*sp*sp
            elseif (ak(4,i).eq.1) then
              t(i)=-t(i)*2.0d0*sp*sp
            endif
20        continue
          lsh=t(1)+t(2)
        endif
        return
        end

C-------------------------------------------------------------
C
C  Subroutine DEFSHP returns the parameters to define the
C  required mode shape derivative.

        subroutine defshp(ix,iy,a1,a2)
        integer*1 a1,a2,lookup,table
        dimension a1(7),a2(7)
        dimension lookup(0:2,0:2),table(14,6)
        data table/1,1,12*0,
     &           1,0,1,0,0,1,8*0,
     &           1,0,1,0,1,9*0,
     &           1,0,0,1,0,2,0,1,0,1,0,2,0,-1,
     &           1,0,0,1,2,0,0,1,0,1,0,0,2,-1,
     &           1,0,0,1,1,1,0,-1,0,1,0,1,1,-1/
        data lookup/1,2,4,3,6,0,5,0,0/

C  These are the powers of the terms

        do 10 i=1,7
          a1(i)=table(i,lookup(ix,iy))
          a2(i)=table(i+7,lookup(ix,iy))
10      continue
        return
        end

C-------------------------------------------------------------
C
C  Subroutine COMREC writes line segment endpoints
C    corresponding to contours at specified levels.
C    This routine translated from BYTE, June 1987.

        subroutine comrec(iub,jub,nc,x,y,z,d)
        implicit real*8 (a-h,o-z)
        dimension x(0:iub),y(0:jub),z(0:nc-1),d(0:50,0:50)
        dimension h(0:4),ish(0:4),xh(0:4)
        dimension yh(0:4),im(0:3),jm(0:3)
        integer*1 castab(0:2,0:2,0:2)
        integer*1 case
        logical*1 prmerr
        data im /0,1,1,0/
```

```
      data jm /0,0,1,1/
      data castab /0,0,9,0,1,5,7,4,8,
     &             0,3,6,2,3,2,6,3,0,
     &             8,4,7,5,1,0,9,0,0/
      one=1.0d0
      prmerr=(iub.le.0.or.jub.le.0.or.nc.le.0)
      if (prmerr) then
        write (*,*) 'Error in parameters passed to COMREC'
        stop
      endif
      do 200 j=jub-1,0,-1
        do 190 i=0,iub-1
          dmin=min(d(i,j),d(i+1,j),d(i,j+1),d(i+1,j+1))
          dmax=max(d(i,j),d(i+1,j),d(i,j+1),d(i+1,j+1))
          if (dmax.lt.z(0).or.dmin.gt.z(nc-1)) goto 190
          do 180 k=0,nc-1
            if (z(k).lt.dmin.or.z(k).gt.dmax) goto 180
            do 20 m=4,0,-1
              if (m.gt.0) then
                ix=i+im(m-1)
                iy=j+jm(m-1)
                h(m)=d(ix,iy)-z(k)
                xh(m)=x(ix)
                yh(m)=y(iy)
              else
                h(0)=(h(1)+h(2)+h(3)+h(4))/4.0d0
                xh(0)=(x(i)+x(i+1))/2.0d0
                yh(0)=(y(j)+y(j+1))/2.0d0
              endif
              if (h(m).gt.0.0) then
                ish(m)=2
              elseif (h(m).lt.0.0) then
                ish(m)=0
              else
                ish(m)=1
              endif
20            continue
              do 150 m=1,4
                m1=m
                m2=0
                m3=mod(m,4)+1
                case=castab(ish(m1),ish(m2),ish(m3))
                if (case.eq.0) goto 150
                if (case.eq.1) then
                  x1=xh(m1)
                  y1=yh(m1)
                  x2=xh(m2)
                  y2=yh(m2)
                elseif (case.eq.2) then
                  x1=xh(m2)
                  y1=yh(m2)
                  x2=xh(m3)
                  y2=yh(m3)
                elseif (case.eq.3) then
```

```
                  x1=xh(m3)
                  y1=yh(m3)
                  x2=xh(m1)
                  y2=yh(m1)
                elseif (case.eq.4) then
                  x1=xh(m1)
                  y1=yh(m1)
                  x2=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
                  y2=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
                elseif (case.eq.5) then
                  x1=xh(m2)
                  y1=yh(m2)
                  x2=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
                  y2=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
                elseif (case.eq.6) then
                  x1=xh(m3)
                  y1=yh(m3)
                  x2=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
                  y2=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
                elseif (case.eq.7) then
                  x1=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
                  y1=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
                  x2=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
                  y2=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
                elseif (case.eq.8) then
                  x1=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
                  y1=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
                  x2=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
                  y2=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
                elseif (case.eq.9) then
                  x1=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
                  y1=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
                  x2=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
                  y2=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
                endif
                write (3,140) x1,y1,x2,y2,z(k)
140              format (1x,5(g10.4,2x))
150             continue
180           continue
190         continue
200       continue
          return
          end

C-------------------------------------------------------------
C
C  Subroutine CONPLT sets up the contour plotting

          subroutine conplt(emin,emax,e)
          implicit real*8 (a-h,o-z)
          dimension e(0:50,0:50),z1(0:10),xc(0:50),yc(0:50)

          write (*,10) emin,emax
10        format (' Strain Levels:'/
```

```
     &          ' Minimum:',g16.8/' Maximum:',g16.8)
15       nc=0
20       write (*,*) 'Enter desired strain contour levels:',
     &           ' [''end'' to exit]'
30       read (*,*,err=32) elevel
         zl(nc)=elevel
         nc=nc+1
         goto 30

32       iflag=0
         do 35 i=0,nc-2
           if (zl(i).gt.zl(i+1)) then
             t=zl(i)
             zl(i)=zl(i+1)
             zl(i+1)=t
             iflag=1
           endif
35       continue
         if (iflag.eq.1) goto 32

         write (*,40) nc
40       format (' Generating',i5,' contours')
         do 50 i=0,25
           xc(i)=xmap(i)
           xc(50-i)=1.0d0-xc(i)
           yc(i)=xc(i)
           yc(50-i)=1.0d0-xc(i)
50       continue
         call comrec(50,50,nc,xc,yc,zl,e)
         return
         end
```

Appendix B: Analytical Results

This appendix contains results from the analytical programs as well as a sample output file.

Analytical Predictions for (0,90) facesheets							
From SPAD, LISA, and DIFA				Damage	Failure		
File	Energy	Velocity	Indent	Width	Stress	Wext	PI
sp..	[ft lbs]	[in/s]	[in]	[in]	[ksi]	[ft lbs]	[ft lbs]
25	0.005	3.703	0.0095	0.0105	40.43	0.0035	0.0066
24	0.010	5.237	0.0124	0.0106	40.43	0.0069	0.0115
26	0.020	7.407	0.0162	0.0106	40.43	0.0134	0.0200
9	0.050	11.711	0.0229	0.1760	33.14	0.0318	0.0421
10	0.100	16.562	0.0297	0.2215	31.18	0.0609	0.0741
8	0.125	18.516	0.0323	0.2629	29.72	0.0752	0.0889
11	0.200	23.422	0.0384	0.2787	29.24	0.1158	0.1310
7	0.250	26.186	0.0417	0.3303	27.89	0.1423	0.1575
12	0.300	28.685	0.0446	0.3370	27.72	0.1684	0.1833
13	0.400	33.123	0.0495	0.3476	27.48	0.2185	0.2329
1	0.500	37.033	0.0537	0.4098	26.25	0.2679	0.2808
14	0.600	40.567	0.0574	0.4172	26.11	0.3164	0.3274
15	0.700	43.818	0.0607	0.4238	26.00	0.3639	0.3728
16	0.800	46.843	0.0637	0.4296	25.90	0.4105	0.4175
17	0.900	49.685	0.0664	0.4348	25.82	0.4554	0.4615
2	1.000	52.372	0.0690	0.5020	24.80	0.5013	0.5049
18	1.250	58.554	0.0748	0.5123	24.74	0.6134	0.6022
3	1.500	64.143	0.0800	0.5212	24.54	0.7256	0.7188
23	1.750	69.282	0.0846	0.5290	24.44	0.8344	0.8247
4	2.000	74.066	0.0889	0.6077	23.50	0.9445	0.9299
5	2.500	82.808	0.0958	0.6198	23.38	1.1386	1.1390
6	3.000	90.711	0.1033	0.6301	23.27	1.3747	1.3465
19	4.000	104.745	0.1150	0.7279	22.35	1.7977	1.7597
20	5.000	117.108	0.1250	0.7418	22.23	2.2143	2.1706
21	6.000	128.285	0.1337	0.7536	22.13	2.6199	2.5807
22	8.000	148.131	0.1489	0.8629	21.25	3.4293	3.3991
27	10.000	165.616	0.1617	0.8789	21.12	4.2144	4.2186
28	15.000	202.837	0.1881	1.0084	20.08	6.1509	6.2758
29	20.000	234.216	0.2092	1.0319	19.89	8.0236	8.3377

Table B.1 Analytical Results for (0,90) Specimens

Analytical Predictions for (±45) facesheets							
From SPAD, LISA, and DIFA				Damage	Failure		
File	Energy	Velocity	Indent	Width	Stress	Wext	PI
dp..	[ft lbs]	[in/s]	[in]	[in]	[ksi]	[ft lbs]	[ft lbs]
25	0.005	3.703	0.0095	0.0000	29.00	0.0035	
24	0.010	5.237	0.0124	0.0000	29.00	0.0069	
26	0.020	7.407	0.0162	0.0000	29.00	0.0134	
9	0.050	11.711	0.0229	0.0000	29.00	0.0318	
10	0.100	16.562	0.0297	0.0000	29.00	0.0609	0.0736
8	0.125	18.516	0.0323	0.0000	29.00	0.0752	
11	0.200	23.422	0.0384	0.0000	29.00	0.1158	
7	0.250	26.186	0.0417	0.2815	26.67	0.1423	
12	0.300	28.685	0.0445	0.2869	26.63	0.1674	
13	0.400	33.123	0.0495	0.2957	26.55	0.2185	
1	0.500	37.033	0.0537	0.3030	26.50	0.2679	0.2784
14	0.600	40.567	0.0573	0.3097	26.44	0.3150	
15	0.700	43.818	0.0606	0.3156	26.39	0.3624	
16	0.800	46.843	0.0636	0.3213	26.34	0.4089	
17	0.900	49.685	0.0663	0.3264	26.30	0.4537	
2	1.000	52.372	0.0689	0.3773	25.88	0.4995	0.5002
18	1.250	58.554	0.0747	0.3883	25.78	0.6113	
3	1.500	64.143	0.0799	0.3972	25.71	0.7233	0.7120
23	1.750	69.282	0.0845	0.4046	25.65	0.8320	
4	2.000	74.066	0.0888	0.4109	25.60	0.9419	0.9209
5	2.500	82.808	0.0965	0.4219	25.50	1.1595	1.1277
6	3.000	90.711	0.1032	0.4320	25.42	1.3714	1.3333
19	4.000	104.745	0.1149	0.4821	25.01	1.7938	
20	5.000	117.108	0.1248	0.5398	24.53	2.2055	2.1480
21	6.000	128.285	0.1336	0.5580	24.38	2.6150	
22	8.000	148.131	0.1487	0.5824	24.18	3.4178	
27	10.000	165.616	0.1617	0.6001	24.03	4.2144	
28	15.000	202.837	0.1879	0.7105	23.11	6.1345	
29	20.000	234.216	0.2089	0.7472	22.81	7.9948	

Table B.2 Analytical Results for (±45) Specimens

Appendix C: Experimental Data

This appendix contains the experimental data in tabular form.

Test Matrix S0	Material Property Tests						
11/16" Panels							
Compression Tests		Failure	Failure		Poisson's		
Specimen ID	Width	Load	Stress	Modulus	Ratio		
	[mm]	[lbs]	[ksi]	[msi]			
E0 -1 -1	75.990	3321.94	37.01	9.3883	0.051		
E0 -1 -2	75.885	4523.69	50.47	8.9082	0.0736		
E0 -1 -3	76.325	4386.91	48.66	9.1221	0.0987		
E45 -1 -1	77.460	2779.68	30.38	2.8633	0.6183		
E45 -1 -2	76.865	2559.84	28.20	3.1373	0.7177		
E45 -1 -3	77.430	2813.88	30.77	3.0277	0.7107		
E90 -1 -1	76.350	4601.86	51.03	8.4839	0.0984		
E90 -1 -2	76.405	4543.24	50.35	9.0331	0.1197		
E90 -1 -3	76.580	4758.18	52.61	8.542	0.0822		
Tension Tests			Failure	Failure	Failure		Poisson's
Specimen ID	Thick	Width	Load	Stress	Strain	Modulus	Ratio
	[mm]	[mm]	[lbs]	[ksi]	*10^-6	[msi]	
D0 -1 -1	0.375	49.77	3527	121.92	12516	9.74	0.11
D0 -1 -2	0.377	49.75	3146	108.22	11770	9.19	0.15
D0 -1 -3	0.361	49.84	3400	121.92	12142	10.04	0.10
D0 -1 -4	0.369	49.90	3180	111.43	12580	8.86	0.03
D0 -2 -1	0.368	48.73	3425	123.20	13632	9.04	0.06
D0 -2 -2	0.371	48.80	3278	116.81	12801	9.13	0.10
D0 -2 -3	0.373	48.68	3127	111.09	10924	10.17	0.13
D0 -2 -4	0.368	48.69	3170	114.16	13797	8.27	0.02
D45 -1 -1	0.383	49.85	708	23.94	9982	2.40	0.69
D45 -1 -2	0.386	49.84	713	23.92	9762	2.45	0.68
D45 -1 -3	0.392	49.82	713	23.56	10044	2.35	0.75
D45 -1 -4	0.392	49.86	713	23.54	9703	2.43	0.75
D45 -2 -1	0.378	48.73	747	26.18	11796	2.22	0.69
D45 -2 -2	0.383	48.84	782	26.96	10710	2.52	0.71
D45 -2 -3	0.397	48.74	775	27.85	10242	2.72	0.73
D45 -2 -4	0.389	48.85	835	28.36	10851	2.61	0.73
D90 -1 -1	0.380	49.78	3146	107.30	11886	9.03	0.05
D90 -1 -2	0.391	49.81	3175	105.19	12296	8.55	0.19
D90 -1 -3	0.391	49.62	3117	103.64	12038	8.61	0.05
D90 -1 -4	0.383	49.89	3493	117.94	13842	8.52	0.04
D90 -2 -1	0.369	48.68	3356	120.54	12851	9.38	0.15
D90 -2 -2	0.365	48.75	3258	118.14	13847	8.53	0.06
D90 -2 -3	0.366	48.76	3200	112.67	12498	9.02	0.17
D90 -2 -4	0.373	48.76	3117	110.56	12903	8.57	0.07

Table C.1 Data From Test Matrix S0

Test Matrix S1-L									
3/8" Panels				Impact	Impact	Damage	Damage	Failure	Failure	
Specimen ID				Energy	Force	Area	Width	Stress	Load	Width
				[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
L0	-1	-1		0.00	0.00			42.90	4465	88.125
L0	-1	-2		0.00	0.00			46.97	4924	88.767
L0	-1	-3		0.00	0.00			49.39	5134	88.018
L0	-2	-1		1.35	156.68			35.85	3737	88.257
L0	-2	-2		1.49	149.56			36.58	3796	87.848
L0	-2	-3		1.80	180.41			32.40	3361	87.837
L0	-3	-1		1.49	149.55			28.95	3004	87.862
L0	-3	-2		1.46	156.68			30.83	3200	87.880
L0	-3	-3								87.333
L0	-4	-1		1.96	140.06			30.30	3156	88.192
L0	-4	-2		1.70	175.67			32.73	3390	87.695
L0	-4	-3		1.85	154.30			29.59	3068	87.783
L0	-5	-1		2.90	161.43			26.94	2794	87.815
L0	-5	-2		3.00	147.18			24.32	2521	87.748
L0	-5	-3		3.00	163.80			24.31	2521	87.805
L0	-6	-1		3.12	161.63			22.94	2384	88.003
L0	-6	-2		3.12	142.43			25.30	2623	87.797
L0	-6	-3								87.973
L45	-1	-1		0.00	0.00	0.000	0.0	27.92	2916	88.435
L45	-1	-2		0.00	0.00	0.000	0.0	27.61	2868	87.930
L45	-1	-3		0.00	0.00	0.000	0.0	29.07	3039	88.495
L45	-2	-1		1.32	163.80		0.0	***23.08***	***2404***	88.175
L45	-2	-2		1.32	159.05	0.067	9.5	***24.72***	***2589***	88.679
L45	-2	-3		1.71	147.18	0.077	10.5	***26.70***	***2780***	88.129
L45	-3	-1		1.26	159.05		0.0	25.59	2667	88.248
L45	-3	-2		1.29	159.05	0.048	8.5	23.93	2496	88.320
L45	-3	-3								88.389
L45	-4	-1		1.46	130.56	0.087	9.0	24.54	2570	88.645
L45	-4	-2		1.49	142.43	0.087	12.0	25.13	2599	87.566
L45	-4	-3		1.49		0.095	11.0	25.22	2623	88.062
L45	-5	-1		2.29	173.30	0.187	12.5	25.38	2653	88.475
L45	-5	-2		2.61	166.17	0.220	13.5	***27.25***	***2848***	88.479
L45	-5	-3		2.22	163.80	0.220	14.0	24.57	2570	88.543
L45	-6	-1		2.22	140.06	0.207	14.5	23.75	2477	88.280
L45	-6	-2		2.61	175.67	0.238	14.0	22.16	2301	87.897
L45	-6	-3								88.415
				Data in ***italic*** represents failure at the grip						
				for damaged speciments. This data was not used.						

Table C.2 Data From Test Matrix S1-L

Test Matrix S1-M									
11/16" Panels				Impact	Impact	Damage	Damage	Failure	Failure	
Specimen ID				Energy	Force	Area	Width	Stress	Load	Width
				[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
M0	-1	-1		1.35				32.403	3385	88.457
M0	-1	-2		1.29	170.92			38.105	3991	88.682
M0	-1	-3						31.205	3273	88.808
M0	-2	-1		2.52	163.80			29.318	3080	88.940
M0	-2	-2		2.61	147.18			25.877	2711	88.710
M0	-2	-3		2.90	149.56			27.057	2843	88.970
M45	-1	-1		1.35	182.79			25.214	2662	89.402
M45	-1	-2		1.62	173.29			26.291	2775	89.358
M45	-1	-3		1.57	170.93			26.224	2746	88.640
M45	-2	-1		2.61	230.27			24.147	2555	89.585
M45	-2	-2		2.80	170.92			25.459	2692	89.517
M45	-2	-3		2.80	166.17			25.316	2662	89.040

Table C.3 Data From Test Matrix S1-M

Test Matrix S1-N									
1" Panels				Impact	Impact	Damage	Damage	Failure	Failure	
Specimen ID				Energy	Force	Area	Width	Stress	Load	Width
				[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
N0	-1	-1		1.80						
N0	-1	-2		2.61	185.16					
N0	-1	-3		4.11	211.28					
N0	-2	-1		1.29	218.40			38.99	4074	88.480
N0	-2	-2		1.66	201.78			39.76	4177	88.953
N0	-2	-3		1.49	175.04			38.54	4074	89.517
N0	-3	-1		1.49	180.42			35.92	3757	88.550
N0	-3	-2		1.49	167.55			35.95	3767	88.713
N0	-3	-3		0.00	0.00				5574	89.303
N0	-4	-1		2.09	175.16			30.75	3195	87.970
N0	-4	-2		1.66	192.29			31.58	3288	88.147
N0	-4	-3		1.53	180.42			38.39	4035	88.987
N0	-5	-1		2.80	175.15			27.42	2873	88.685
N0	-5	-2		2.80	201.78			31.36	3273	88.365
N0	-5	-3		2.80	211.27			28.49	3014	89.585
N0	-6	-1		3.00	194.66			29.12	3039	88.348
N0	-6	-2		2.61	200.19			27.87	2902	88.145
N0	-6	-3		0.00	0.00				4954	88.308
N45	-1	-1		0.00	0.00	0.000	0.0	29.88	3146	89.145
N45	-1	-2		0.00	0.00	0.000	0.0	20.70	2188	89.472
N45	-1	-3		0.00	0.00	0.000	0.0	30.49	3224	89.525
N45	-2	-1		1.17	154.30	0.073	9.0	29.46	2985	85.794
N45	-2	-2		1.10	149.56	0.067	8.0	27.68	2917	89.206
N45	-2	-3		1.01	159.05	0.057	7.0	28.17	2990	89.868
N45	-3	-1		1.35	161.43	0.090	9.0	26.49	2794	89.315
N45	-3	-2		1.17	161.43	0.083	9.0	26.49	2794	89.297
N45	-3	-3								89.835
N45	-4	-1		1.75	182.79	0.130	12.0	26.20	2736	88.406
N45	-4	-2		1.75	192.29	0.097	8.0	26.40	2760	88.529
N45	-4	-3		1.71	175.67	0.150	11.5	26.08	2755	89.435
N45	-5	-1		2.61	180.42	0.240	15.0	25.44	2658	88.463
N45	-5	-2		2.70	173.30	0.283	15.0	24.38	2550	88.575
N45	-5	-3		2.90	156.68	0.317	16.0	23.58	2457	88.239
N45	-6	-1		2.80	144.81	0.303	15.0	24.41	2555	88.626
N45	-6	-2		2.80	154.30	0.317	16.0	24.47	2565	88.747
N45	-6	-3								88.370

Table C.4 Data From Test Matrix S1-N

Test Matrix S2-L											
3/8" Panels				Damage	Damage	Impact	Impact	Damage	Damage	Failure	Failure	
Specimen ID				Type	Level	Energy	Force	Area	Width	Stress	Load	Width
						[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
L0	-7	-1		Static	.130 in		173.49	0.093	10.5	32.98	3468	89.045
L0	-7	-2		Static	.130 in		187.19	0.100	12.0	30.95	3263	89.280
L0	-7	-3		Static	.130 in		168.93	0.100	13.0	28.65	3029	89.500
L0	-8	-1		Static	.244 in		164.36	0.310	17.0	28.67	3009	88.870
L0	-8	-2		Static	.244 in		168.93	0.303	15.0	26.08	2750	89.300
L0	-8	-3		Static	.244 in		159.80	0.280	16.5	23.80	2501	88.990
L0	-9	-1		Dynamic	40 mm	1.62	144.81	0.063	11.0	34.55	3669	89.918
L0	-9	-2		Dynamic	43 mm	1.97	149.56	0.073	13.0	32.67	3444	89.268
L0	-9	-3		Dynamic	50 mm	3.00	185.16	0.197	17.0	26.25	2770	89.350
L45	-7	-1		Static	.130 in		150.66	0.107	9.0	24.73	2599	88.970
L45	-8	-3		Static	.130 in		159.80	0.113	15.0	24.65	2599	89.250
L45	-7	-3		Static	.130 in		187.19	0.120	11.0	25.48	2692	89.440
L45	-8	-1		Static	.244 in		173.49	0.317	15.5	23.95	2521	89.110
L45	-8	-2		Static	.244 in		173.49	0.287	16.0	23.27	2457	89.390
L45	-7	-2		Static	.244 in		191.75	0.287	10.0	24.03	2540	89.510

Table C.5 Data from Test Matrix S2-L

Test Matrix S2-M											
11/16" Panels				Damage	Damage	Impact	Impact	Damage	Damage	Failure	Failure	
Specimen ID				Type	Level	Energy	Force	Area	Width	Stress	Load	Width
						[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
M0	-3	-1		Static	.119 in		187.19	0.060	9.0	34.51	3625	88.925
M0	-3	-2		Static	.119 in		173.49	0.063	10.5	33.86	3566	89.178
M0	-3	-3		Static	.119 in		187.19	0.070	11.5	34.92	3674	89.073
M0	-20	-1		Static	.240 in		155.23	0.280	16.0	24.83	2638	89.970
M0	-20	-2		Static	.240 in		178.06	0.290	18.0	25.72	2726	89.718
M0	-20	-3		Static	.240 in		168.93	0.285	16.5	27.26	2873	89.213
M0	-21	-1		Dynamic	40 mm	1.07	149.55	0.045	10.0	37.03	3889	88.920
M0	-21	-2		Dynamic	43 mm	1.42	159.05	0.080	13.5	36.15	3810	89.255
M0	-21	-3		Dynamic	50 mm	2.09	159.05	0.185	14.0	25.44	2677	89.092
M45	-20	-1		Static	.119 in		168.93	0.085	8.5	28.48	3029	90.037
M45	-20	-2		Static	.119 in		159.80	0.095	11.0	28.26	3004	90.018
M45	-20	-3		Static	.119 in		173.49	0.085	9.0	28.56	3029	89.777
M45	-6	-1		Static	.240 in		205.45	0.285	13.0	26.03	2711	88.203
M45	-6	-2		Static	.240 in		178.06	0.270	16.0	25.46	2662	88.538
M45	-3	-1		Static	.240 in		191.77	0.265	15.0	25.08	2653	89.538

Table C.6 Data from Test Matrix S2-M

Test Matrix S2-N												
1" Panels				Damage	Damage	Impact	Impact	Damage	Damage	Failure	Failure	
Specimen ID				Type	Level	Energy	Force	Area	Width	Stress	Load	Width
						[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
N0	-20	-1		Static	.108 in		150.67	0.040	9.0	39.051	4113	89.182
N0	-20	-2		Static	.108 in		146.10	0.085	11.0	37.001	3894	89.093
N0	-20	-3		Static	.108 in		150.67	0.080	11.0	34.367	3635	89.542
N0	-21	-1		Static	.236 in		150.67	0.280	17.5	29.335	3083	88.968
N0	-21	-2		Static	.236 in		210.02	0.250	15.0	25.200	2653	89.123
N0	-21	-3		Static	.236 in		164.36	0.265	16.0	25.984	2736	89.142
N0	-22	-1		Dynamic	40 mm	1.35	159.05	0.115	11.0	37.852	3991	89.275
N0	-22	-2		Dynamic	43 mm	1.66	149.56	0.140	13.0	35.722	3762	89.155
N0	-22	-3		Dynamic	50 mm	2.61	159.05	0.250	16.5	24.466	2599	89.937
N45	-20	-1		Static	.108 in		173.49	0.080	8.0	29.148	3063	88.972
N45	-20	-2		Static	.108 in		173.49	0.070	8.0	28.710	3004	88.600
N45	-20	-3		Static	.108 in		168.93	0.075	7.5	29.383	3087	88.965
N45	-21	-1		Static	.236 in		182.62	0.305	15.0	26.553	2789	88.943
N45	-21	-2		Static	.236 in		191.76	0.280	14.5	25.455	2667	88.718
N45	-21	-3		Static	.236 in		196.32	0.280	15.0	26.626	2789	88.700

Table C.7 Data from Test Matrix S2-N

Test Matrix S3									
11/16" Panels				Impact	Impact	Damage	Damage	Failure	Failure	
Specimen I			Level	Energy	Force	Area	Width	Stress	Load	Width
				[ft lbs]	[lbs]	[in^2]	[mm]	[ksi]	[lbs]	[mm]
C0	-1	-1	None	0.00				54.89	8690	89.358
C0	-1	-2	None	0.00				53.82	8510	89.242
C0	-1	-3	None	0.00				51.34	8140	89.495
C0	-2	-1	Low	1.26	218.40	0.02	6.00	37.94	5990	89.110
C0	-2	-2	Low	1.05	216.02	0.00	0.00	40.71	6430	89.155
C0	-2	-3	Low	1.10	220.77	0.01	3.00	40.79	6470	89.538
C0	-3	-1	Low	1.07	197.03	0.04	6.00	40.78	6460	89.403
C0	-3	-2	Low	1.15	201.78	0.04	6.00	37.85	6000	89.478
C0	-4	-1	Medium	1.53	230.27	0.07	8.00	35.03	5560	89.578
C0	-4	-2	Medium	1.42	194.66	0.09	6.50	37.73	5990	89.600
C0	-4	-3	Medium	1.42	189.91	0.08	8.00	36.35	5760	89.453
C0	-5	-1	High	2.29	223.15	0.20	12.00	28.16	4440	88.997
C0	-5	-2	High	2.29	230.27	0.13	9.00	33.55	5300	89.168
C0	-5	-3	High	2.36	x	0.15	11.00	35.00	5530	89.185
C0	-6	-1	High	2.22	225.52	0.17	10.50	34.94	5530	89.323
C0	-6	-2	High	2.29	201.78	0.13	9.00	33.99	5370	89.177
C45	-1	-1	None	0.00				54.53	8620	89.225
C45	-1	-2	None	0.00				54.65	8660	89.452
C45	-1	-3	None	0.00				54.03	8540	89.223
C45	-2	-1	Low	0.97	178.04	0.00	0.00	41.86	6630	89.405
C45	-2	-2	Low	1.01	x	0.01	2.00	39.63	6260	89.165
C45	-2	-3	Low	1.05	170.92	0.02	6.00	41.50	6580	89.493
C45	-3	-1	Low	1.10	185.16	0.00	0.00	40.75	6440	89.205
C45	-3	-2	Low	1.20	199.41	0.01	3.00	40.92	6460	89.108
C45	-4	-1	Medium	1.46	199.41	0.06	9.00	38.57	6090	89.132
C45	-4	-2	Medium	1.42	189.91	0.05	7.00	40.12	6340	89.198
C45	-4	-3	Medium	1.42	206.53	0.09	6.50	41.31	6540	89.355
C45	-5	-1	High	2.15	218.40	0.13	10.00	37.90	5970	88.910
C45	-5	-2	High	1.97	254.01	0.12	13.00	37.19	5880	89.243
C45	-5	-3	High	2.15	204.16	0.13	11.50	38.04	6050	89.763
C45	-6	-1	High	2.29	201.78	0.12	12.50	34.78	5490	89.087
C45	-6	-2	High	2.22	239.76	0.11	11.00	36.09	5720	89.448

Table C.8 Data from Test Matrix S3