Automatic Meshing of CAD Ship Files for Use with Numerical Electromagnetics Codes

by

David C. Jenn

July 1996

Approved for public release; distribution is unlimited.

Prepared for:  Commander, Space and Naval Warfare Systems Command
Code PMW 163
Arlington, VA 22245-5200
NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral M.J. Evans
Superintendent

R. Elster
Provost

This report was prepared for and funded by SPAWAR PMW163.
Reproduction of all or part of this report is authorized.

The report was prepared by:

David C. Jenn
DAVID C. JENN
Associate Professor
Department of Electrical and
Computer Engineering

Reviewed by:

Herschel H. Loomis, Jr.
Chairman
Department of Electrical and
Computer Engineering

Released by:

David W. Netzer
Associate Provost and
Dean of Research
1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
   July 15, 1996

3. REPORT TYPE AND DATES COVERED
   October 1, 1994 to July 1, 1996

4. TITLE AND SUBTITLE
   Automatic Meshing of CAD Ship Files for Use with Numerical Electromagnetics Codes

5. FUNDING NUMBERS

6. AUTHOR(S)
   David C. Jenn

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
   Naval Postgraduate School
   Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
   NPS-EC-96-012

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
   SPAWAR PMW163
   2451 Crystal Park 5
   Arlington, VA 22245-5200

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
   The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
   Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
   Computer aided design (CAD) is routinely employed in the structural and mechanical design of ships and aircraft. With relatively minor modifications, the same geometry databases can be used by computational electromagnetics (EM) codes for antenna and radar cross section analysis. A step-by-step procedure is described to obtain triangular facet models derived from CAD data files. The facet models can be used by EM codes such as PATCH and NEC. Computer code listings of translation software and instructions for use are included.

14. SUBJECT TERMS
   Computational electromagnetics, CAD

15. NUMBER OF PAGES
   98

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
   UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
   UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
   UNCLASSIFIED

20. LIMITATION OF ABSTRACT
   SAR
Table of Contents

1.0 INTRODUCTION 1
2.0 CAD PROGRAMS 2
  2.1 ACAD GENERAL DESCRIPTION 2
  2.2 AUTOCAD GENERAL DESCRIPTION 3
  2.3 IGES FILES 5
    2.3.1 INTRODUCTION 5
    2.3.2 IGES FOR AUTOCAD 6
    2.3.3 IGES FOR ACAD 7
  2.4 ACAD GENERIC FACET FILE 12
  2.5 SHIP CAD FILES 15
3.0 PATCH 15
  3.1 PATCH GENERAL DESCRIPTION 15
  3.2 PATCH INPUT FILE FORMAT 21
  3.3 PATCH CODE MODIFICATIONS 25
    3.3.1 INPUT/OUTPUT 26
    3.3.2 FACET CHECKING SUBROUTINE 26
4.0 VIEWING GEOMETRY FILES 29
  4.1 INTRODUCTION 29
  4.2 VIEW USING BUILDN5 30
  4.3 VIEW USING ACAD 30
  4.4 VIEW USING MATLAB 30
5.0 TRANSLATORS AND COMPUTER CODES 30
  5.1 INTRODUCTION 30
  5.2 ACAD-TO-PATCH TRANSLATOR 31
  5.3 PATCH-TO-ACAD TRANSLATOR 31
  5.4 PATCH-TO-NEC TRANSLATOR 31
  5.5 FILE CHECKING USING KNIT 33
6.0 AUTOMESHING PROCEDURE 33
REFERENCES 37
APPENDICES
  APPENDIX A: FACET CHECKING CODE 38
  APPENDIX B: ACAD-TO-PATCH TRANSLATOR CODE 44
  APPENDIX C: PATCH-TO-ACAD TRANSLATOR CODE 56
  APPENDIX D: PATCH-TO-NEC TRANSLATOR CODE 64
  APPENDIX E: PATCH INPUT CHECKING CODE (KNIT) 66
  APPENDIX F: GEOMETRY FILE BUILDER (BLDMAT) 78
  APPENDIX G: MATLAB GEOMETRY VIEWER (PLTPATCH) 89
INITIAL DISTRIBUTION LIST 92
1.0 INTRODUCTION

Computer aided design (CAD) has become a standard tool used both by the Navy and contractors in the design of ships. Detailed ship drawings are generated with CAD software such as AutoCAD. The advantage is that modification and redesign are relatively easy because of the associativity of entities (i.e., points, lines, circles, surfaces, etc.). The CAD database contains the relationships between all entities. Therefore, the secondary effects of modifying an edge, for example, are automatically accounted for when other structures that contain the modified edge are generated.

The ability for rapidly modifying a structure is ideal for concurrent engineering, that is, the simultaneous design of a system by all engineering disciplines. Electromagnetic (EM), fluid, and structural designers can all use the same CAD database for their analyses. Any changes made to the system are immediately available to all engineers.

For the electromagnetic design of systems, several computer codes are available to government agencies and contractors that use triangular facet models of bodies to define scattering and radiation structures. They include PATCH, CARLOS-3D, and FERM [1-3], all of which are written in the FORTRAN language. FERM has its own preprocessing program to generate geometry files. Furthermore, it uses binary data files which are not easily transportable between computer codes and systems. CARLOS-3D can use geometry files generated by the CAD program ACAD without modification. However, CARLOS-3D can only solve scattering problems, not radiation (antenna) problems. PATCH is capable of solving both scattering and radiation problems. It uses an ASCII input file that defines the body on the basis of edge connections (as opposed to triangle connections). The edge definition approach can be ambiguous in some isolated (but predictable) special cases.

For the primary application considered here, high frequency (HF) ship antenna analysis, PATCH has been found to be most useful. However, PATCH has only a basic geometry preprocessor capable of simple shapes such as plates, cylinders, cones, and spheres. The data for more complex shapes must be input by hand. It involves defining the nodes of every triangle on the body and specifying an edge connection list. Ship models that are meshed for use up to 30 MHz have about 3000 nodes (triangle vertices) and 5000 edges. In this case, data entry itself is a major effort. Furthermore, if a ship modification is to be investigated, portions of the structure may have to be remeshed and re-input, which again represents a major effort. It is apparent that for computational EM codes to be practical engineering tools, an efficient surface meshing operation must be available.
The CAD application ACAD (Advanced Computer Aided Design) is capable of performing the desired automeshing. Databases from other CAD programs can be imported into ACAD and then meshed and output in a special "facet" format. The facet file is ASCII and contains the node and facet information required by PATCH, although it is not in the proper format. An ACAD-to-PATCH translator was written to reformat the facet file into one that is recognized by PATCH. Therefore, it is now possible to take a NAVSEA CAD file, mesh it automatically in ACAD, and then use the output to run PATCH.

This report summarizes the development of the automeshing process and the step-by-step procedure to go from a CAD drawing file to PATCH input file. First an overview is given of the two CAD applications used here: ACAD and AutoCAD. The automeshing procedure is described, and special file format translation software is also presented. Finally, some shipboard applications are discussed and a few helpful hints and guidelines are given.

2.0 CAD PROGRAMS

2.1 ACAD GENERAL DESCRIPTION

ACAD (Advanced Computer Aided Design) [4-5] provides users with the ability to create and modify geometry in two- or three-dimensions. Users can choose to model geometry with wireframes, surfaces, or solids. ACAD is the primary tool used by Lockheed Fort Worth Company's Advanced Programs for configuration and subsystem design of new and existing aircraft programs. ACAD's primary role is the generation of geometry and some limited analysis. Much of the analysis performed within ACAD is geometrical analysis. For other types of analysis, ACAD generates interface files for transferring to groups who specialize in a particular analysis field such as Radar Cross Section (RCS), Aero, or Computational Fluid Dynamics (CFD).

Inputting data to ACAD is accomplished through one of many input modes available to the designer. Example options include digitizing locations, entering explicit coordinate values, snap to grid, and intersections. Each entity (splines, lines, points, surfaces, etc.) can have individual color, width, and style attributes. Logical groupings of entities can be separated and managed with layers, groups, and blanking. ACAD models can be viewed orthographically or in perspective. Users can specify view orientation and choose to display geometry in multiple window configurations. Window operations such as panning, zooming, and auto extents are accomplished at any time providing instream capability. The ACAD user can also control the display

\[1\] Most of the material in this section is taken directly from reference 4

2
of surfaces or solids with options as wireframe, hidden line removal, flat, or Gouraud shading.

At the heart of the ACAD system is the associative database. In an associative database, geometry is linked together in a relational structure that remembers parent/child dependencies. This type of database enables rapid modifications of geometry, since modifying one geometric element automatically adjusts its dependencies based on a set of predefined rules. For instance, changing a control spline of a fuselage will automatically regenerate any surface(s) built with the spline. In turn, any geometry that is associated to the fuselage surface (i.e., plane/curve and surface intersections, fillets) will automatically regenerate.

Within ACAD exists a read/write Initial Graphics Exchange Specification (IGES) translator. The IGES translator allows ACAD the ability to exchange drawing data with other CAD systems that support IGES. Example CAD systems include CAM, CATIA, COMPUTERVISION, and AutoCAD.

The ACAD system also supports a binary data file converter. This built in converter enables a binary file created on the SUN to be read in directly on a Silicon Graphics or Apollo workstation without having to convert to a neutral ASCII file. This utility is extremely beneficial to projects supporting a mixture of workstations and using a transparent networking system. Hardcopy output is available on a variety of devices accessed through ACAD. Additional features of the system are:

1. Three-dimensional lines drawings
2. Analysis geometry models
3. Three- and Five-View drawings

Table 1 contains more information on the types of commands available on ACAD Version 9.0.

2.2 AUTOCAD GENERAL DESCRIPTION

The capabilities of AutoCAD\(^3\) [6] are similar to those of ACAD. AutoCAD is one of the most widely distributed CAD programs, and runs on all platforms (UNIX, DOS and Windows, and Macintosh). As in the case of ACAD, the effect of every change made to a drawing appears immediately on screen.

---

\(^2\)Most of the material in this section is taken directly from reference 6

\(^3\)Frequently the AutoCAD suite of programs is collectively referred to as ACAD. This terminology is not used here to avoid confusion with the ACAD discussed in Section 2.1.
<table>
<thead>
<tr>
<th>Transformations</th>
<th>Intersections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>Curve-Curve</td>
</tr>
<tr>
<td>Translate</td>
<td>Curve-Plane</td>
</tr>
<tr>
<td>Rotate</td>
<td>Curve-Surface</td>
</tr>
<tr>
<td>Mirror</td>
<td>Plane-Surface</td>
</tr>
<tr>
<td>Copy</td>
<td>Surface-Surface</td>
</tr>
<tr>
<td>1024 User Defined Layers</td>
<td>Curve Projections onto Surfaces</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display Options</th>
<th>Drafting Utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank On, Off, and On Only</td>
<td>Break, Trim, and Join Curves</td>
</tr>
<tr>
<td>Color, Style, and Width Line Fonts</td>
<td>Corner</td>
</tr>
<tr>
<td>Hidden Line, Flat, and Gouraud Shading</td>
<td>Grouping</td>
</tr>
<tr>
<td>Auxiliary Viewing</td>
<td>Construction Planes</td>
</tr>
<tr>
<td>Orthographic or Perspective Viewing</td>
<td>Local Coordinate Systems</td>
</tr>
<tr>
<td>Multiple Windows (up to 6)</td>
<td>Offsets</td>
</tr>
<tr>
<td>Dynamic Viewing</td>
<td>Text and Dimensions</td>
</tr>
<tr>
<td>Zooming, Panning, and Auto Extents</td>
<td>Groups, Dittos, and Details</td>
</tr>
<tr>
<td></td>
<td>Crosshatching</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Options</th>
<th>Three Dimension Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digitize</td>
<td>Point, Line, and Spline Primitives</td>
</tr>
<tr>
<td>Reference Existing Data Points</td>
<td>Conic, Circles, and Ellipses Primitives</td>
</tr>
<tr>
<td>Key in Explicit z, y, z</td>
<td>Six Forms of Surfaces</td>
</tr>
<tr>
<td>Intersection</td>
<td>Curve and Surface Editing</td>
</tr>
<tr>
<td>Point On</td>
<td>Trimmed Surfaces (Faces)</td>
</tr>
<tr>
<td>Snap to Grid</td>
<td>Mass Properties (volumes, CGS, areas)</td>
</tr>
<tr>
<td>Hierarchical Input Mode</td>
<td>Offset Surfaces</td>
</tr>
<tr>
<td></td>
<td>Wireframe, Surface, &amp; Solid Modeling</td>
</tr>
</tbody>
</table>

Table 1: Summary of ACAD Commands
AutoCAD functions lets the user modify the drawing in a variety of ways. Entities can be erased or moved, or copied to form repeated patterns. The user can change the view of the drawing displayed on screen, or display information about the drawing. AutoCAD also provides drawing aids that allow the positioning of entities accurately. The simple command format of AutoCAD allows the user to accomplish most of the functions in Table 1. One important exception is the inability to generate a shell mesh. Therefore, automeshing must be performed in ACAD.

2.3 IGES FILES

2.3.1 INTRODUCTION

The AutoCAD and ACAD support translation of drawings to and from the Initial Graphics Exchange Specification (IGES). IGES is a public-domain data specification intended as an international standard for the exchange of information between CAD systems. An IGES translator is written specifically for a given CAD program but, in principle, enables drawings to be transferred to and from other CAD systems that also support IGES (in IGES, these are known as sending and receiving systems).

Files created using the IGES format will consist of at most six sections of information, five of which are mandatory. These sections must exist in the following order:

1. Flag Section (not always present)

   This section of the IGES file signals the format used to write the file. The absence of this section is interpreted to mean that the normal ASCII format was used when creating the file.

2. Start Section

   The Start Section is intended to be a human-readable prologue containing comments about the IGES file.

3. Global Section

   The Global Section of the IGES file contains information about the CAD system that created the file and information that should be considered by the CAD system interpreting the file before the IGES file is processed.

4. Directory Entry Section

   The Directory Entry Section consists of one two-line entry for each entity described by the IGES file.

---

4For a complete description of IGES, see reference 7.
5. Parameter Data Section

The Parameter Data Section of the IGES file contains the geometric information that will be used to reconstruct each entity.

6. Terminate Section

The goal of an IGES translation is to preserve the geometry and functionality of entities in a CAD drawing or an IGES file. This process does have limits. As with all translation, concepts that can be expressed precisely in one language may not have exact equivalents in another language; conversely, concepts common to two languages may be expressed differently by each. The situation applies as much to CAD data as to natural languages. When entities have no direct correspondence between IGES and the entities in a particular CAD program, the translator maps them to similar constructs that attempt to preserve as much data as possible.

For example, IGES has no direct equivalent to an AutoCAD tapered polyline. To translate a tapered polyline, AutoCAD creates an IGES Composite Curve segment whose width is the average of the polyline’s starting and ending widths. Similarly, AutoCAD has no counterpart to the IGES Parametric Spline Surface, so AutoCAD approximates these entities with 3D meshes.

Many drawings can be translated with little or no loss of data, but even in this case, the entities used to represent the drawing may change in translation. This means that IGES is not fully symmetrical: reading an IGES file with the same program that was used to create it does not necessarily lead to a drawing that is identical to the original. The more complex the drawing, the more likely that information will have to be approximated (this applies especially to drawings that are heavily annotated or hierarchically organized, or that use complex three-dimensional entities). For a one-time translation to or from IGES, this may not pose a great problem. If, however, one is concerned with maintaining drawings that must be translated between the two systems over a period of time, it is necessary to be familiar with the details of both the send and receive translators as well as the details of IGES formats.

IGES has been distributed in successive versions, with each version providing additional features and enhancements to existing features. Because it is intended as a long-term standard, a new IGES version attempts to support all features that have been officially part of any earlier version.

2.3.2 IGES FOR AUTOCAD

The AutoCAD IGESIN and IGESOUT commands support translation of drawings to and from IGES. AutoCAD generates files that are compatible with IGES 4.0.
It can successfully read files that conform to the IGES 2.0, 3.0, and 4.0 file formats provided they employ the fixed-length ASCII form. IGES entities supported by AutoCAD are shown in Figures 1 and 2.

2.3.3 IGES FOR ACAD

ACAD has the ability to transfer a drawing to other systems by creating a file having the IGES format. Limitations on the transfer of drawings are a result of the limitations of the IGES translators of both the sending and receiving systems. The ACAD/IGES translator is used to transfer drawings between ACAD and other systems such as CADAM, CATIA and AutoCAD. The ACAD/IGES translator was created using the methodology described [7].

Entities which are supported by the current version of the ACAD/IGES translator are listed in Figures 3 and 4. Several points should be noted when using the ACAD system to translate drawings to and from the IGES file format:

1. When writing an IGES file from ACAD, entities which are confined to a single plane are represented as an IGES entity constructed in the XY plane with an associated transformation matrix to translate the entity to the appropriate plane.

2. Spline entities written from ACAD into the IGES format can be represented as either IGES type 112 (Parametric Spline Curve) or type 126 (Rational B-Spline Curve) depending on the selection of the buttons under the “Spline Types:” section of the “Write IGES File” dialog box. If the button “Parametric” is selected, all spline entities will be written as type 112; otherwise, all spline entities will be written as type 126.

3. The method in which a Surface entity is written into IGES format from the ACAD system will depend on the setting of the “Surface Type:” buttons in the “Write IGES File” dialog box.

4. When writing ACAD Face entities into the IGES format, several IGES entities are created. The Face entity is represented as type 144 (Trimmed (Parametric) Surface) which consists of boundary curves that are represented by type 142 (Curve on a Parametric Surface). Boundary curves, in turn, are created through type 102 (Composite Curve), which links together simple entities such as points, lines, circles, splines, conics, and ellipses, forming a closed curve.
<table>
<thead>
<tr>
<th>Code</th>
<th>Data Type</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(Null)</td>
<td>Supported</td>
</tr>
<tr>
<td>100</td>
<td>(Circular Arc)</td>
<td>Supported</td>
</tr>
<tr>
<td>102</td>
<td>(Composite Curve)</td>
<td>Supported</td>
</tr>
<tr>
<td>104</td>
<td>(Conic Arc)</td>
<td>Supported</td>
</tr>
<tr>
<td>106</td>
<td>(Copious Data)</td>
<td>Supported</td>
</tr>
<tr>
<td>108</td>
<td>(Plane)</td>
<td>Supported</td>
</tr>
<tr>
<td>110</td>
<td>(Line)</td>
<td>Supported</td>
</tr>
<tr>
<td>112</td>
<td>(Parametric Spline Curve)</td>
<td>Supported</td>
</tr>
<tr>
<td>114</td>
<td>(Parametric Spline Surface)</td>
<td>Supported</td>
</tr>
<tr>
<td>116</td>
<td>(Point)</td>
<td>Supported</td>
</tr>
<tr>
<td>118</td>
<td>(Ruled Surface)</td>
<td>Supported</td>
</tr>
<tr>
<td>120</td>
<td>(Surface Of Revolution)</td>
<td>Supported</td>
</tr>
<tr>
<td>122</td>
<td>(Tabulated Cylinder)</td>
<td>Supported</td>
</tr>
<tr>
<td>124</td>
<td>(Transformation Matrix)</td>
<td>Supported</td>
</tr>
<tr>
<td>125</td>
<td>(Flash)</td>
<td>Not supported</td>
</tr>
<tr>
<td>126</td>
<td>(Rational B-Spline Curve)</td>
<td>Supported</td>
</tr>
<tr>
<td>128</td>
<td>(Rational B-Spline Surface)</td>
<td>Not supported</td>
</tr>
<tr>
<td>130</td>
<td>(Offset Curve)</td>
<td>Not supported</td>
</tr>
<tr>
<td>132</td>
<td>(Connect Point)</td>
<td>Not supported</td>
</tr>
<tr>
<td>134</td>
<td>(Node)</td>
<td>Not supported</td>
</tr>
<tr>
<td>136</td>
<td>(Finite Element)</td>
<td>Not supported</td>
</tr>
<tr>
<td>138</td>
<td>(Nodal Displacement &amp; Rotation)</td>
<td>Not supported</td>
</tr>
<tr>
<td>140</td>
<td>(Offset Surface)</td>
<td>Not supported</td>
</tr>
<tr>
<td>142</td>
<td>(Curve On A Parametric Surface)</td>
<td>Not supported</td>
</tr>
<tr>
<td>144</td>
<td>(Trimmed Parametric Surface)</td>
<td>Not supported</td>
</tr>
<tr>
<td>146</td>
<td>(Nodal Result)</td>
<td>Not supported</td>
</tr>
<tr>
<td>148</td>
<td>(Element Results)</td>
<td>Not supported</td>
</tr>
<tr>
<td>150</td>
<td>(Block)</td>
<td>Not supported</td>
</tr>
<tr>
<td>152</td>
<td>(Right Angular Wedge)</td>
<td>Not supported</td>
</tr>
<tr>
<td>154</td>
<td>(Richt Circular)</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

**Figure 1:** IGES to AutoCAD conversion table.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>302</td>
<td>(Associativity Definition) Not supported.</td>
</tr>
<tr>
<td>304</td>
<td>(Line Font Definition) Output by IGESOUT if necessary. Partially supported by IGESIN.</td>
</tr>
<tr>
<td>306</td>
<td>(MACRO Definition) Not supported.</td>
</tr>
<tr>
<td>308</td>
<td>(Subfigure Definition) Supported.</td>
</tr>
<tr>
<td>310</td>
<td>(Text Font Definition) Not supported.</td>
</tr>
<tr>
<td>312</td>
<td>(Text Display Template) Not supported.</td>
</tr>
<tr>
<td>314</td>
<td>(Color Definition) Not supported.</td>
</tr>
<tr>
<td>320</td>
<td>(Network Subfigure Definition) Not supported.</td>
</tr>
<tr>
<td>322</td>
<td>(Attribute Table Definition) Not supported. Instead of creating Attribute Table Definitions, IGESOUT translates AutoCAD Attributes into a pair consisting of a General Note and a Property entity, and attaches these as Subfigure references (see section 2.2.4.4.2 of the IGES 5.1, and the section &quot;Attributes&quot; on page 18).</td>
</tr>
<tr>
<td>402</td>
<td>(Associativity Instance) IGESOUT supports Forms 1, 3, 4, 7, 13, 14, 15, and 16. IGESOUT supports Form 3.</td>
</tr>
<tr>
<td>404</td>
<td>(Drawing) Supported.</td>
</tr>
<tr>
<td>406</td>
<td>(Property) Partial support: see the section &quot;Attributes&quot; on page 18.</td>
</tr>
<tr>
<td>408</td>
<td>(Singular Subfigure Instance) Supported.</td>
</tr>
<tr>
<td>410</td>
<td>(View) Supported.</td>
</tr>
<tr>
<td>412</td>
<td>(Rectangular Array Subfigure Instance) Supported, except that IGESIN does not support the DO-DON'T flags.</td>
</tr>
<tr>
<td>414</td>
<td>(Circular Array Subfigure Instance) Not supported.</td>
</tr>
<tr>
<td>416</td>
<td>(External Reference) Supported. See the section &quot;External References&quot; on page 23 and &quot;External Reference — 416&quot; on page 40.</td>
</tr>
<tr>
<td>418</td>
<td>(Nodal Load/Constraint) Not supported.</td>
</tr>
<tr>
<td>420</td>
<td>(Network Subfigure Instance) Not supported.</td>
</tr>
<tr>
<td>422</td>
<td>(Attribute Table Instance) Not supported.</td>
</tr>
<tr>
<td>430</td>
<td>(Solid Instance) Not supported.</td>
</tr>
<tr>
<td>600-699</td>
<td>(MACRO Instance) Not supported.</td>
</tr>
<tr>
<td>5001-9999</td>
<td>(Implementor-Defined) IGESOUT uses form 7901 for Block Attributes.</td>
</tr>
<tr>
<td>10000-9999</td>
<td>(MACRO Instance) Not supported.</td>
</tr>
</tbody>
</table>

Figure 2: IGES to AutoCAD conversion table (continued).
<table>
<thead>
<tr>
<th>IGES Entity Number</th>
<th>IGES Description</th>
<th>ACAD Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Arc</td>
<td>Arc/Circle</td>
</tr>
<tr>
<td>102</td>
<td>Composite Curve</td>
<td>any combination of: point, line, circle spline,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conic, offset</td>
</tr>
<tr>
<td>104 Form 0–3</td>
<td>Conic Arc</td>
<td>Conic/Ellipse</td>
</tr>
<tr>
<td>106 Form 1,2,11,12</td>
<td>Copious Data</td>
<td>any combination of: point, line</td>
</tr>
<tr>
<td>20,21,31–38,40,63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>108 Form 0</td>
<td>Plane</td>
<td>Plane</td>
</tr>
<tr>
<td>110</td>
<td>Line</td>
<td>Line</td>
</tr>
<tr>
<td>112</td>
<td>Parametric Spline</td>
<td>Spline</td>
</tr>
<tr>
<td>114</td>
<td>Parametric Spline Surface</td>
<td>Arbitrary Surface</td>
</tr>
<tr>
<td>116</td>
<td>Point</td>
<td>Point</td>
</tr>
<tr>
<td>118 Form 1</td>
<td>Ruled Surface</td>
<td>Ruled Surface</td>
</tr>
<tr>
<td>120</td>
<td>Surface of Revolution</td>
<td>Surface of Revolution</td>
</tr>
<tr>
<td>122</td>
<td>Tabulated Cylinder</td>
<td>Ruled Surface</td>
</tr>
<tr>
<td>124 Form 0</td>
<td>Transformation Matrix</td>
<td>N/A</td>
</tr>
<tr>
<td>126 Form 0</td>
<td>Rational B–Spline Curve</td>
<td>Spline</td>
</tr>
<tr>
<td>128 Form 0</td>
<td>Rational B–Spline Surface</td>
<td>Arbitrary Surface</td>
</tr>
<tr>
<td>130</td>
<td>Offset Curve</td>
<td>Spline</td>
</tr>
<tr>
<td>142</td>
<td>Curve on a Parametric Surf.</td>
<td>any combination of: point, line, circle, spline,</td>
</tr>
<tr>
<td></td>
<td>only sub—entity of 144</td>
<td>conic, offset</td>
</tr>
<tr>
<td>144</td>
<td>Trimmed (Param.) Surface</td>
<td>Face</td>
</tr>
<tr>
<td>202</td>
<td>Angular Dimension</td>
<td>Angular Dimension</td>
</tr>
<tr>
<td>206</td>
<td>Diameter Dimension</td>
<td>Diameter Dimension</td>
</tr>
<tr>
<td>210</td>
<td>General Label</td>
<td>Label Dimension</td>
</tr>
<tr>
<td>212 Form 0,2–8</td>
<td>General Note</td>
<td>Text</td>
</tr>
<tr>
<td>214 Form 2,4</td>
<td>Leader (Arrow)</td>
<td>Arrow Dimension</td>
</tr>
<tr>
<td>216</td>
<td>Linear Dimension</td>
<td>Vertical/Horizontal/Parallel Dimension</td>
</tr>
<tr>
<td>218</td>
<td>Ordinate Dimension</td>
<td>Vertical Call Out/Horizontal Call Out/Label</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dimension</td>
</tr>
<tr>
<td>220</td>
<td>Point Dimension</td>
<td>Vertical Call Out/Horizontal Call Out/Label</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dimension</td>
</tr>
<tr>
<td>222</td>
<td>Radius Dimension</td>
<td>Radial Dimension</td>
</tr>
<tr>
<td>308</td>
<td>Subfigure Definition</td>
<td>Detail</td>
</tr>
<tr>
<td>402 Form 1,3,4,7,</td>
<td>Associativity Instance</td>
<td>Views, Sets</td>
</tr>
<tr>
<td>14,15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>Drawing</td>
<td>Parent View (limit 3 per file)</td>
</tr>
<tr>
<td>406 Form 15</td>
<td>Name</td>
<td>Names Details</td>
</tr>
<tr>
<td>408</td>
<td>Singular Subfigure Instance</td>
<td>Ditto</td>
</tr>
<tr>
<td>410</td>
<td>View</td>
<td>Auxiliary View</td>
</tr>
</tbody>
</table>

Figure 3: IGES to ACAD conversion table.
<table>
<thead>
<tr>
<th>ACAD Entity</th>
<th>IGES Entity Number</th>
<th>IGES Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point</td>
<td>116</td>
<td>Point</td>
</tr>
<tr>
<td>Line</td>
<td>110</td>
<td>Line</td>
</tr>
<tr>
<td>Circle/Arc</td>
<td>100</td>
<td>Arc *</td>
</tr>
<tr>
<td>Spline</td>
<td>112</td>
<td>Parametric Spline or</td>
</tr>
<tr>
<td></td>
<td>126 Form 0</td>
<td>Rational B—Spline Curve</td>
</tr>
<tr>
<td>Conic</td>
<td>104 Form 0,3</td>
<td>Conic Arc *</td>
</tr>
<tr>
<td>Ellipse</td>
<td>104 Form 1</td>
<td>Conic Arc *</td>
</tr>
<tr>
<td>Plane</td>
<td>108 Form 0</td>
<td>Plane</td>
</tr>
<tr>
<td>Surface Identities:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PolyConic, Fillet</td>
<td>114</td>
<td>Parametric Spline Surface or</td>
</tr>
<tr>
<td></td>
<td>128 Form 0</td>
<td>Rational B—Spline Surface</td>
</tr>
<tr>
<td>Ruled</td>
<td>118 Form 1</td>
<td>Ruled Surface or</td>
</tr>
<tr>
<td></td>
<td>114</td>
<td>Parametric Spline Surface or</td>
</tr>
<tr>
<td></td>
<td>128 Form 8</td>
<td>Rational B—Spline Surface</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>Surface of Revolution or</td>
</tr>
<tr>
<td>Surface of Revolution</td>
<td>114</td>
<td>Parametric Spline Surface or</td>
</tr>
<tr>
<td></td>
<td>128 Form 0</td>
<td>Rational B—Spline Surface</td>
</tr>
<tr>
<td>Sculptured, Arbitrary, Offset</td>
<td>114</td>
<td>Parametric Spline Surface or</td>
</tr>
<tr>
<td></td>
<td>128 Form 0</td>
<td>Rational B—Spline Surface</td>
</tr>
<tr>
<td>Face</td>
<td>144</td>
<td>Curve on a Param. Surf. and</td>
</tr>
<tr>
<td></td>
<td>142</td>
<td>Trimated (Param) Surf. and</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>Composite Curve</td>
</tr>
<tr>
<td>Solids:</td>
<td>Not Implemented</td>
<td></td>
</tr>
<tr>
<td>Polyhedral</td>
<td>Not Implemented</td>
<td></td>
</tr>
<tr>
<td>Volume</td>
<td>Not Implemented</td>
<td></td>
</tr>
<tr>
<td>Primitive</td>
<td>Not Implemented</td>
<td></td>
</tr>
<tr>
<td>Annotation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>212 Form 0,6,7,8</td>
<td>General Note</td>
</tr>
<tr>
<td>Dimensions:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Horizontal, Vertical, Parallel</td>
<td>216</td>
<td>Linear Dimension %</td>
</tr>
<tr>
<td>Angular</td>
<td>202</td>
<td>Angular Dimension %</td>
</tr>
<tr>
<td>Radial</td>
<td>222</td>
<td>Radius Dimension %</td>
</tr>
<tr>
<td>Diameter</td>
<td>206</td>
<td>Diameter Dimension %</td>
</tr>
<tr>
<td>Label, Textline</td>
<td>210</td>
<td>General Label %</td>
</tr>
<tr>
<td>Arrow</td>
<td>214 Form 2,4</td>
<td>Leader (Arrow)</td>
</tr>
<tr>
<td>Station Label</td>
<td>218</td>
<td>Ordinate %</td>
</tr>
<tr>
<td>View Call Out</td>
<td>110</td>
<td>Lines and</td>
</tr>
<tr>
<td></td>
<td>212</td>
<td>General Note</td>
</tr>
<tr>
<td>Structures:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detail</td>
<td>308</td>
<td>Subfigure Definition *</td>
</tr>
<tr>
<td>Ditto</td>
<td>408</td>
<td>Singular Subfigure Instance</td>
</tr>
<tr>
<td>Aux Views</td>
<td>410</td>
<td>View</td>
</tr>
<tr>
<td>Parent View</td>
<td>404</td>
<td>Drawing</td>
</tr>
</tbody>
</table>

* written in the XY plane with attached Transformation Matrix (124 Form 0).

% entity contains pointers to the corresponding General Note (212), Witness Lines (106 Form 40), Arrows (214), and Transformation Matrix (124 Form 0).

Figure 4: IGES to ACAD conversion table (continued).
2.4 ACAD GENERIC FACET FILE

The ACAD Generic Facet file is an ASCII file containing faceted (polygonal) data as well as edge, vertex, and surface normal data. The files are named with the "facet" suffix. The convention for surface normals is to point away from the interior of the part. The triangular facet shape is the only one currently supported. The Generic Facet format version V3.0 is described below:

Line A: Revision Date/Time Machine

Revision = File format version;

Date/Time = Date and time of creation;

Machine = Hardware platform of origin.

Line B: NP

NP = Number of parts in file.

Line C: Part Name

Part Name = Name of current part.

Line D: MIRROR (A B C D)

MIRROR = Mirrored about a plane?

(if not mirrored about a plane then MIRROR = 0 and A, B, C, D, are not present)

Line E: NV

NV = Number of vertices in current part.

Line F: X Y Z

(X Y Z) = 3-D Cartesian point:

(there will be NV copies of this line).

Line G: NSP

NSP = Number of subparts in current part.
Line H: Sub-Part Name

Sub-Part = Name of the current subpart.

Line J: ET NSE NSV EM2 VP VN EC

ET = Type of element (Triangle = 3);
NSE = Number of elements in current subpart — always ≠ 0;
NSV = Number of vertices in the current subpart ≠ 0 if VP = 1 or VN = 1;
EM2 = Parameter set to 1 if 2-sided material fields have been defined, and 0 otherwise:

When set to 1, Line N will have 3 material fields, instead of 1. In these fields, M will describe the material associated with the element only, M1 will describe the material associated with the plus normal side of the element and M2 the material associated with the minus normal side of the elements.

VP = Parameter set to 1 if vertex parameters are present, and 0 otherwise:

Line K is present in the file only when this parameter is set to 1.

VN = Parameter set to 1 if vertex normals are present, and 0 otherwise:

Line L is present in the file only when this parameter is set to 1.

EC = Parameter set to 1 if element curvature lines are present, and 0 otherwise:

Line M is present in the file only when this parameter is set to 1.

Line K: U V VID

(U V) = Parametric vertex coordinate — there will be NSV copies of this line if VP = 1;

VID = Vertex ID referencing the part vertex list.

Line L: Nx Ny Nz VID

(Nx Ny Nz) = 3-D unit vector pointing away from the interior of the part:

(there will be NSV Copies of this line if VN = 1)

VID = Vertex ID referencing the part vertex list.
**Line M:** Min Max MnVx MnVy MnVz MxVx MxVy MxVz

Min/Max = The principal curvatures computed at the facet center

(positive values indicate surface bending towards the normal)

MnVx MnVy MnVz = A 3-D unit vector pointing in the direction where curvature is a minimum;

MxVx MxVy MxVz = Points in the direction of maximum curvature.

(MnV × MxV = N)

(there will be NSE copies of this line if EC = 1)

**Line N:** V1 V2 M

V1, V2 = Indices of subpart vertices if NSV ≠ 0, otherwise indices of part vertices;

M = Fields to describe properties associated with this element

(there will be NE copies of this line if ET = 2)

**Line N:** V1 V2 V3 M (M1 M2)

V1, V2, V3 = Indices of subpart vertices if NSV ≠ 0, otherwise indices of part vertices;

M (M1 M2) = Fields to describe material properties associated with this element

(there will be NE copies of this line if ET = 3)

**Line N:** V1 V2 V3 V4 M (M1 M2)

V1, V2, V3, V4 = Indices of subpart vertices if NSV ≠ 0, otherwise indices of part vertices;

M (M1 M2) = Fields to describe material properties associated with this element

(there will be NE copies of this line if ET = 4)

**Line N:** V1 V2 V3 V4 V5 V6 M (M1 M2)

V1, V2, V3, V4, V5, V6 = Indices of subpart vertices if NSV ≠ 0, otherwise indices of part vertices;
M (M1 M2) = Fields to describe material properties associated with this element

(there will be NE copies of this line if ET = 6)

Sections H thru N are repeated for each subpart.

Sections C thru N are repeated for each part.

2.5 SHIP CAD FILES

NAVSEA and its contractors have made extensive use of three-dimensional ship drawings. Engineers have their personal preferences when it comes to CAD software, and therefore the IGES file format is extremely important in the exchange of databases between platforms and programs. AutoCAD binary files for the ships listed in Table 2 were provided to NPS by NAVSEA. The files were imported to AutoCAD residing on NPS platforms, and then translated to IGES using the IGESOUT command. Next, the output file is read into ACAD, and an ACAD binary file written for future structure modification. Also, an ACAD facet file can be generated for translation to PATCH input format.

Three-dimensional ship drawings generated by AutoCAD are shown in Figures 5 through 7 for a DDG-51, LHA, and LHD, respectively. Figure 8 illustrates the level of detail that is typical of the ship databases. When performing EM computations at HF this level of detail is not required; only structures on the order of 0.1\( \lambda \) need to be represented. (At the high end of the HF band the frequency is 30 MHz and the wavelength 10 meters.) Furthermore, meshing the surfaces of small structures introduces a large additional number of edges (i.e., unknowns) that do not improve the quality of the solution, but dramatically increases the computer run time. Therefore, it is necessary to filter the geometry model to remove unnecessary detail. This step can be done either in AutoCAD (before the IGES file is written), or after the model has been imported to ACAD.

3.0 PATCH

3.1 PATCH GENERAL DESCRIPTION

PATCH [1] is a FORTRAN computer code that computes electromagnetic scattering and radiation based on a method of moments (MM) solution of the E-field integral equation (EFIE). The method of moments reduces the EFIE to a set of linear equations that can be solved using standard matrix methods. The number of unknowns, and hence the size of the matrix equation that must be solved, depends
Figure 5: AutoCAD generated model of a DDG-51 (without weapons systems and antennas).
Figure 6: AutoCAD generated model of a LHA.
Figure 7: AutoCAD generated model of a LHD (no hull file was available).
Figure 8: AutoCAD generated model of a LHA forward mast platform.
Table 2: Ship 3D CAD Files

<table>
<thead>
<tr>
<th>Ship</th>
<th>Imported in ACAD</th>
<th>Meshed in ACAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDG51</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LHA</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LHD*</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DD963</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

* No hull file

on the number of triangular patches that are used to represent the scattering body.

The method of moments is rigorous; that is, in the limit as the triangles become smaller, the method of moments solution converges to the correct value. Unlike a wire simulation (wire grid model), the area between edges which form the triangle facets are solid material, not air gaps. Therefore, current truely flows on the surface of the object, not just along the edges of the triangles.

For an object with $N$ edges, PATCH computes a vector of complex coefficients $I_m$, $m = 1, 2, \ldots, N$, such that the current crossing edge $m$ is

$$\vec{J}_m = I_m \hat{n}_m.$$

The unit vector $\hat{n}_m$ is in a direction normal to edge $m$ and lies on the surface. Once the current coefficients have been determined using the method of moments procedure, it is possible to compute radiation patterns and scattered fields. A summary of the capabilities of PATCH is given in Table 3. The details of modeling the deckedge antennas are discussed in [8].

The application of MM is usually limited by the size of the computer available. Bodies comprised of large numbers of triangles yield matrices too large for the com-
Table 3: Summary of PATCH Capabilities

<table>
<thead>
<tr>
<th><strong>Arbitrary Shape</strong></th>
<th><strong>Excitation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open/Closed objects</td>
<td>Voltage sources (e.g., for antennas)</td>
</tr>
<tr>
<td>Modelled by triangular “patches”</td>
<td>Plane waves</td>
</tr>
<tr>
<td>Variable patch density</td>
<td>Both</td>
</tr>
<tr>
<td>Front end for graphical composition</td>
<td></td>
</tr>
<tr>
<td><strong>Arbitrary edge multiplicity</strong></td>
<td><strong>Calculated Quantities</strong></td>
</tr>
<tr>
<td></td>
<td>Surface currents</td>
</tr>
<tr>
<td></td>
<td>Far field patterns</td>
</tr>
<tr>
<td></td>
<td>Radar cross sections</td>
</tr>
<tr>
<td></td>
<td>Field calculations at general observation points (including near field points)</td>
</tr>
<tr>
<td></td>
<td><strong>Frequency Domain</strong></td>
</tr>
<tr>
<td>Non-orientable surfaces (e.g., Moebius strip) OK</td>
<td>Pattern loops</td>
</tr>
<tr>
<td>Symmetry planes may be included</td>
<td>Frequency loops</td>
</tr>
<tr>
<td>Multiple bodies okay</td>
<td></td>
</tr>
</tbody>
</table>

The computer, or run times too long to be of practical use. A general rule of thumb for convergence of the far field is that triangle edge lengths should not exceed 0.1λ, where λ is the wavelength. The number of edges is generally a close estimate of the number of unknowns that must be determined. Differences are due to the fact that some edges may be shared by more than two facets, and therefore the number of current coefficients associated with that edge is more than two. (This is referred to in the PATCH manual as the multiplicity of the edge.) A SGI Indigo II workstation with 128 MBytes of memory can handle structures with approximately 6000 edges.

3.2 PATCH INPUT FILE FORMAT

The PATCH input file is an ASCII file that contains all of the geometry information and calculation parameters. The file can be generated using the preprocessing code BUILDN5 which is distributed along with PATCH. BUILDN5 is capable of generating basic geometrical shapes and combining them to yield more complex shapes. In addition to geometry information, BUILDN5 also prompts the user for calculation information such as frequency, observation angles, and excitation conditions.

The user input data is appended to the geometry file and written in a format that is recognizable to PATCH. Upon execution, the NPS version of PATCH (which has been modified from the original) looks for a file named “inpatch” in the current directory. It performs the required calculations and generates an ASCII output file.
named "outpatch" which contains all of the input data in readable form. An output file of current coefficients is also generated.

It is not necessary that BUILDN5 be used to generate the input file. If the user is familiar with the required file format, "inpatch" can be generated using any text editor. The input data format follows:

**Line A:** TITLE

TITLE = 80 character ASCII string.

**Line B:** NNODES,NEDGES

NNODES = Number of nodes in file.

NEDGES = Number of edges in file.

**Line C:** NODE, X(NODE), Y(NODE), Z(NODE)

NODE = Node index.

X = X coordinate of node number NODE.

Y = Y coordinate of node number NODE.

Z = Z coordinate of node number NODE.

(there will be NNODES copies of this line)

**Line D:** NEDGE, NODE1(NEDGE), NODE2(NEDGE)

NEDGE = Edge number.

NODE1 = Node number of first end.

NODE2 = Node number of second end.

(there will be NEDGES copies of this line; the order of endpoints is not important)

**Line E:** IGNDP(1), IGNDP(2), IGNDP(3)

---

5This input sequence is typical for the calculation of antenna patterns or received signals due to plane wave incidence. It does not necessarily cover all possible input sequences. See reference 1 for a complete discussion of the input format.
IGNDP = Symmetry planes at x = 0, y = 0 or z = 0?

0 = no ground plane; 1= infinite PMC; -1 = infinite PEC

**Line F:** NEXCIT

NEXCIT = Number of voltage excitations.

**Line G:** ITYPE

ITYPE = Type of excitation.

p = plane wave; v= voltage; b = both

(there are NEXCIT copies of this line)

**Line H:** If ITYPE = p: THETA, PHI, ETRE, ETIM, EPRE, EPIM

THETA, PHI = (θ, φ) angle of incidence in polar coordinates.

ETRE, ETIM = Real(£θ), Imag(£θ).

EPRE, EPIM = Real(£φ), Imag(£φ).

**Line I:** IMAG(1), IMAG(2), IMAG(3)

IMAG = Image the plane wave about x = 0, y = 0, or z = 0?

0 = no image; 1= image for PMC; -1 = image for PEC

**Line H:** If ITYPE ≠ p: NVOLT

NVOLT = Number of voltage sources.

**Line I:** IEDGV, IPOS, VREAL, VIMAG

IEDGEV = Edge number for face on which the voltage source resides.

IPOS = Node opposite IEDGEV for positive voltage sense.

VREAL = Real part of impressed voltage.

VIMAG = Imaginary part of impressed voltage.

(there will be NVOLT copies of this line)

**Line J:** NFZS
NFZS = Number of faces with nonzero surface impedance.

**Line K:** IFZS, ZSRE, ZSIM

IFZS = Face number.

ZSRE = Real part of surface impedance.

ZSIM = Imaginary part of surface impedance.

(there will be NFZS copies of this line)

**Line L:** THEV, MTHEV, IETHEV

THEV = Thevinin equivalent circuit?

If THEV = .false. then MTHEV = 0 and IETHEV = 0

If THEV = .true. then MTHEV = basis function index

If THEV = .true. then IETHEV = edge index

**Line M:** IPATT

IPATT = 0, no pattern calculation.

IPATT = 1, pattern calculation with 3-point integration.

IPATT = 2, pattern calculation with 1-point integration.

**Line N:** If IPATT ≠ 0: TH1, TH2, NTH, PH1, PH2, NPH

TH1, TH2 = θ pattern limits

PH1, PH2 = ϕ pattern limits

NTH, NPH = number of pattern points in θ and ϕ

**Line O:** NNFLD

IPATT = Number of field observation points.

**Line P:** If NNFLD ≠ 0: DX, DY, DZ

DX, DY, DZ = Finite difference increments in x, y, z.

**Line Q:** NNFLD ≠ 0: RFLD(1,J), RFLD(2,J), RFLD(3,J)
RFLD(1-3,J) = x, y, z coordinates of field point j
(there will be NNFLD copies of this line)

Line R:  PRINTC
         IPATT = .true., print a current table.

Line S:  NEDGO
         NEDGO = number of edges in current table.

Line T:  If NEDGO ≠ 0 NEGCUR
         NEGCUR = Edges in current table.
(there will be NEDGO copies of this line)

Line U:  FREQ or -1
         FREQ = Frequency.
(terminates when FREQ = -1)

3.3 PATCH CODE MODIFICATIONS

The PATCH source code received from Sandia Labs has been modified to provide additional capabilities that were not available in the original version. Thus PATCH actually refers to a collection of codes. The particular versions of interest for this application are:

1. PATCH2V

   There are no significant changes between this version and the original one provided by Sandia. The major change is the addition of “facet checking” as described below.

2. PATCHDF

   This version has been modified to specifically compute the current induced by incident plane waves at ship deckedge antenna locations. The edge indices corresponding to the deckedge antenna locations must be provided. The induced currents for all specified incidence angles are written to a ASCII file for use by the programs RECAL and RMSDF. This version also does “facet checking.”
Figure 9: An example of an ambiguous edge-defined surface.

3.3.1 INPUT/OUTPUT

All versions of PATCH run at NPS generate a set of MATLAB ".m" files. The files contain the computed field quantities, and can be loaded into MATLAB for plotting. An ASCII file of the computed current coefficients named "currents" is also generated. This is a duplicate of the list that would occur in "outpatch" if the user requested a listing of the currents.

3.3.2 FACET CHECKING SUBROUTINE

PATCH defines geometrical shapes on the basis of edge connections; i.e., it searches for three connected edges and considers the enclosed edges to define a unique facet. Figure 9 illustrates a situation that occurs frequently in the meshing of surfaces that PATCH misinterprets. PATCH finds four triangles (ABD, ACD, CDB, and ABC) when in fact there are only three (ABD, ACD, and CDB). Therefore, before declaring that a face has been found, the new face should be checked against all previously defined faces to see if they have any common area.

An efficient test based on a comparison of circles inscribed inside of the two triangles under consideration is illustrated in Figure 10. If the inscribed circles overlap then the two facets share area and the larger of the two triangles is not a valid face. This test can fail in the case of extreme aspect ratios as shown in Figure 11. However, the "Shell Mesh" parameters can be set to avoid this situation.
Two triangles need to be tested only if:

1. they share a common edge, and

2. they lie in the same plane\textsuperscript{6}.

Referring to Figure 10, the triangle nodes are \( P_{ij} \) where the first subscript denotes the triangle number \( (i = 1, 2) \) and the second the node number \( (j = 1, 2, 3) \). Similarly the edges are defined by \( \vec{L}_{ij} \), which in vector form are

\[
\vec{L}_{i1} = (x_{i2} - x_{i1})\hat{x} + (y_{i2} - y_{i1})\hat{y} + (z_{i2} - z_{i1})\hat{z} \\
\vec{L}_{i2} = (x_{i3} - x_{i2})\hat{x} + (y_{i3} - y_{i2})\hat{y} + (z_{i3} - z_{i2})\hat{z} \\
\vec{L}_{i3} = (x_{i1} - x_{i3})\hat{x} + (y_{i1} - y_{i3})\hat{y} + (z_{i1} - z_{i3})\hat{z}
\]

Position vectors to the nodes are:

\[
\vec{R}(P_{ij}) = x_{ij}\hat{x} + y_{ij}\hat{y} + z_{ij}\hat{z}
\]

The perimeter of triangle \( i \) is

\[
C_i = \sum_{j=1}^{3} |\vec{L}_{ij}|
\]

\textsuperscript{6}Unless they form the open end of a pyramid or corner. This condition will not be encountered in general.
Figure 11: An example of a case in which the test fails.

In terms of the node position vectors the area of triangle \( i \) is given by

\[
A_i = |\vec{R}(P_{i1}) \times \vec{R}(P_{i2}) + \vec{R}(P_{i2}) \times \vec{R}(P_{i3}) + \vec{R}(P_{i3}) \times \vec{R}(P_{i1})|
\]

A normal vector is the cross product of any two edges. For instance,

\[
\vec{N}_i = \vec{L}_{i1} \times \vec{L}_{i2}
\]

Finally, the position vector to the center of the inscribed circle for triangle \( i \) is

\[
\vec{O}_i = \left[ \frac{\vec{L}_{i2} |\vec{R}(P_{i1})| + \vec{L}_{i3} |\vec{R}(P_{i2})| + \vec{L}_{i1} |\vec{R}(P_{i3})|}{C_i} \right]
\]

The radius of the circle is \( r_i = A_i / C_i \). Two circles \( i \) and \( j \) overlap if the distance between their centers is less than the sum of their radii

\[
|\vec{O}_i - \vec{O}_j| < r_i + r_j
\]

Using the above equations, the following test can be applied to determine whether or not a triangle is a valid face:

1. Loop through all pairs of triangles. For two triangles that share an edge,
2. see if they lie in the same plane (to within some tolerance).
3. If they do, find out which triangle has the smallest area (smallest inscribed circle) and find the location of the center.
4. Determine if the circles overlap by finding the distance between center and comparing it to the sum of the radii.

5. If the circles overlap the large triangle is not a valid face.

The facet checking algorithm has been incorporated into all of the PATCH related codes that generate a face list from the edges. They include:

- All versions of PATCH (new subroutine GEOM and additional subroutines AXB and FACETCK)
- PATCH-to-ACAD translator, PTA
- Input file checking program, KNIT
- MATLAB plot file generating program, BLDMAT

The required changes to the PATCH codes to implement facet checking are:

- new subroutine GEOM
- additional subroutine AXB (vector cross product)
- additional subroutine FACETCK

These codes are listed in Appendix A. Note that the argument list in GEOM is not the same as that for the original version.

4.0 VIEWING GEOMETRY FILES

4.1 INTRODUCTION

Geometry files can be viewed using several different methods. The information that can be displayed differs in each case. The options available are:

1. DISSPLA graphics via the program BUILDN5 (Facet, node, and edge numbers can all be displayed.)

2. ACAD (Only facet numbers are displayed. Edge and node numbers can be found using the edge connection list generated by the program BLDMAT.)

3. MATLAB (Only node and edge numbers can be displayed.)
The procedures required to use each of the above three methods are described below.

4.2 VIEW USING BUILDN5

A PATCH input file can be viewed on the screen or printed using a postscript file by running BUILDN5. The “disp” option is chosen after the data file has been read. The program prompts for the quantities to be displayed (nodes, edges, or faces), the limits of the viewing box, and viewing angle. The DISSPLA software package is required for this method.

4.3 VIEW USING ACAD

Geometries can be viewed directly in ACAD after they have been meshed. However, if facet numbers in “inpatch” are to be the same as those viewed in ACAD, it is necessary to convert the PATCH input file back to the “.facet” form using PATCH’s index ordering. This is achieved using the PATCH-to-ACAD translator PTA. PTA creates a the file “out.facet” from an input file “in.patch” in which each face defined by PATCH is written as a separate part. Therefore, facet numbers assigned by PATCH will be the same as the facet numbers viewed in ACAD using the “Verify Entity” command.

It is not possible to find edge numbers visually in ACAD with the current translator software. Edge and node numbers can be found indirectly by noting the two face numbers that are attached to the edge. The edge number can be determined by finding the common edge in the connection list generated by PATCH or BLDMAT.

4.4 VIEW USING MATLAB

The geometry can be displayed using MATLAB. First the geometry file (in PATCH format) is converted to a set of “.m” files using the FORTRAN program BLDMAT. After BLDMAT has been executed, the MATLAB script PLTPATCH can be run. PLTPATCH has several flags that control whether a wire grid, surface, or surface with hidden lines is displayed. Flags can also be set to display edge and node numbers. The scale and view can be changed using the standard MATLAB commands.

5.0 TRANSLATORS AND COMPUTER CODES

5.1 INTRODUCTION

Throughout the course of this research several new computer codes were written to perform data translation and manipulation. The bulk of the translator codes are
simply subroutines that have been extracted from PATCH, with some minor modifications. The functions and relationships between the various computer codes are illustrated in Figure 12. (Names with a "x" extension refer to executable files rather than the FORTRAN source codes which carry a "f" extension.)

Among the codes developed are the following translators:

1. ACAD to PATCH
2. PATCH to ACAD
3. PATCH to NEC

NEC (Numerical Electromagnetics Code) is a computational EM code based on wire grid models [9].

5.2 ACAD-TO-PATCH TRANSLATOR

The ACAD-to-PATCH translator ATP converts an ACAD "facet" file (described in Section 2.4) to a PATCH input file (described in Section 3.2). A listing of program ATP is given in Appendix B. The ACAD file defines the geometry using a facet table which contains the node coordinates for each triangle. ATP defines edges for each triangle and forms an edge connection list. It uses the same algorithm that is used in PATCH's subroutine GEOM. Therefore, facet checking using the inscribed circle method is also incorporated into ATP.

5.3 PATCH-TO-ACAD TRANSLATOR

The PATCH-to-ACAD translator PTA converts a PATCH input file into ACAD's "facet" format. There are actually two versions of this translator: PTA and PTF. PTA writes the entire ship as a single part. Therefore, when loaded into ACAD, the individual triangles cannot be manipulated (i.e., deleted, moved, verified, etc.). PTF writes an ACAD "facet" file where each facet is a part. PTF results in a much larger file than PTA. Listings are given in Appendix C.

5.4 PATCH-TO-NEC TRANSLATOR

NEC is relatively old compared to the EM patch codes and therefore more widely distributed. A crude translator was written so that a PATCH input format file could be run on NEC. The translator maps each facet edge to a wire segment. This can cause problems because circular wires are usually represented by thin strips in patch codes. Thus the two edges of a thin strip result in two closely spaced parallel wires that may share common space if the wire radii are large enough. This problem is
Figure 12: Computer code functions and relationships.
probably not severe enough to cause NEC to abort, but will affect the computed current on the offending edges.

5.5 FILE CHECKING USING KNIT

The program KNIT reads a PATCH format file and checks it for duplicate edges and nodes. This can occur when ACAD creates a body of revolution. For example, a cylinder can be created by rotating one line about a second line. This is analogous to wrapping a sheet of paper to form a cylinder. Where the two paper edges meet, two lines are created by ACAD. The presence of two overlapping edges can possibly cause problems when running PATCH and should be eliminated. KNIT removes the duplicate quantities and shifts the indices of subsequent entries in the node and edge tables down by one.

6.0 AUTOMESHING PROCEDURE

The following steps are used to generate a triangular facet model of a structure that has been created in AutoCAD using ACAD's "Shell Mesh" command. In the following discussion it is assumed that the reader has a basic knowledge of ACAD commands.

Step 1: Create an IGES file of the database in AutoCAD using IGESOUT.

Step 2: Import the IGES file into ACAD using "Read Other Format" under the "File" box.

Step 3: After successfully reading the IGES file, save the data using the binary option under the "File" menu. The file name should have a "a9" extension. This file will serve as a backup in the event that ACAD crashes during one of the subsequent steps.

Step 4: Modify the file to suit the current problem objectives. This may involve discarding entities that contain too much detail. Examples are window frames, stairways, and internal deckhouse structures such as shelves. Overlapping surfaces must also be eliminated.

Step 5: Once the ship surface has been uniquely defined by non-overlapping surfaces, create faces on all surfaces. The "Include Surface Boundaries" option should be used.

Step 6: Create a "Sheet Assembly" from the faces in Step 5. It is recommended that a large minimum edge length be used on the first attempt to create the
assembly. Also, “Cluster” and “Deviation” parameter values of 50% are recommended. Note that the “Facegap” parameter will impact the final continuity of the surface; i.e., whether adjacent faces share a common edge or a gap exists between them. The continuity of the surface can be checked using the “Verify Laminar Edges” option.

**Step 7:** Once the shell mesh has been generated and the solid volume defined, turn off (blank) all entities except the volume. Save the volume in a “.facet” file.

**Step 8:** Copy the “.facet” file from Step 7 to a file named “in.facet.” Run the ACAD-to-PATCH translator, ATP. A PATCH format file is written to “out.patch.”

**Step 9:** The program KNIT can be run to check the file “out.patch” for duplicate nodes and edges.

**Step 10:** Add the calculation parameters to the geometry file. This can be done using a text editor (if the user is familiar with the PATCH input file format), or by running BUILDN5.

**Step 11:** Copy the file obtained in Step 10 to one named “inpatch” and execute the desired version of PATCH.

Here are several useful hints that should save time reduce errors:

1. The body surfaces (usually of type “Ruled” or “Net”) should be displayed as curved mesh using U and V grid parameters of 2. This allows the surfaces to be viewed and their integrity verified without undue cluttering. Before creating the faces in Step 5, the U and V grid parameters can be changed to 1. Note that in many cases the “Fit Tol” parameter can be important.

2. Individual surfaces can be verified using the “Verify” command with the “Amb” toggle switched on. Triangles that have only two flashing edges must be deleted and recreated, or the endpoints flipped on one of the edges. (Caution: this affects the integrity of any facet attached to the edge being flipped.)

3. The shell meshing (Step 6) should be performed on moderately sized subsections of a large complex target, rather than attempting to mesh the entire structure.

Figures 13 and 14 show a DD963 that has been meshed in ACAD using the above procedure. Remeshing the surface to obtain a larger or smaller grid is the relatively
Figure 13: DD963 ship model generated using the method described in Section 6.
Figure 14: DD963 ship model remeshed with a small grid size.
simple part of the procedure (which starts at Step 6).

REFERENCES


APPENDIX A: FACET CHECKING CODE

The following subroutines perform the facet checking algorithm described in Section 3.3.2

```
c==============================================================================
subroutine geom(datnod,nconn,nedges,itrad,nbound,mxface,
               $ nfacs,nmbnd,numknb)
c==============================================================================

c MODIFIED TO DO FACET CHECKING -- NOTE ARRAY datnod IS REQUIRED

c this subroutine fills nbound with the faces formed by the edges in

c nconn. it also fills in the multiplicity factor of the edge in

c nconn(3,edge). it returns the number of faces(nfaces), and the number

c of body unknowns(numknb) which is equal to the summation of the

c multiplicity factors of the edges before symmetry planes are considered.

c itrak is a work array.

   integer nconn(3,nedges),nbound(3,mxface),itrak(nedges),np(6)
dimension datnod(3,mxmbnd)
nfaces=0

c find faces and list them in nbound.
do 100 iedge=1,nedges-2
    ntrak=0

c look for all edges that attach to edge iedge and put them in itrak.
do 200 jedge=iedge+1,nedges
   do 21 i=1,2
      do 20 j=1,2
         if(nconn(i,iedge).eq.nconn(j,jedge))then

c we have found an edge.

            ntrak=ntrak+1
            itrak(ntrak)=jedge
            goto 200
         endif
21      continue
20      continue
200     continue

c find all pairs of edges that form a face with iedge.
do 300 jedge=1,ntrak-1
   do 301 kedge=jedge+1,ntrak
      do 30 j=1,2
         do 31 k=1,2

c if the 2 faces in itrak have a common point and

c the common point is not in common with the iedge...

            if((nconn(j,itrak(jedge)).eq.nconn(k,itrak(kedge))).and.
               $(nconn(j,itrak(jedge)).ne.nconn(1,iedge)).and.nconn(j,itrak(jedge))
               $.ne.nconn(2,iedge))then
               if(nfaces.eq.0) then

nc if this is the first face save it

               nfacs=nfaces+1
```

38
c put the face into nbound.
nbound(1,nfaces)=iedge
nbound(2,nfaces)=itrak(jedge)
nbound(3,nfaces)=itrak(kedge)
c increment the multiplicity factor of the edges.
nconn(3,iedge)=nconn(3,iedge)+1
nconn(3,itrak(jedge))=nconn(3,itrak(jedge))+1
nconn(3,itrak(kedge))=nconn(3,itrak(kedge))+1
go to 311
def

c****************************************************************************************************************************************
c if this is not the first face, check to see if it overlaps with any
 c previously found face.
k1=iedge
k2=itrak(jedge)
k3=itrak(kedge)
np(1)=nconn(1,k1)
np(2)=nconn(2,k1)
np(3)=nconn(1,k2)
np(4)=nconn(2,k2)
np(5)=nconn(1,k3)
np(6)=nconn(2,k3)
c find the three unique points
node1=np(1)
node2=np(2)
do 603 ii=3,6
npt=np(ii)
   if((npt.ne.np(1)).and.(npt.ne.np(2))) then
   c must be the third node
   node3=npt
go to 602
endif
603 continue
602 continue
c node coordinates of the first face
x11=datnod(1,node1)
y11=datnod(2,node1)
z11=datnod(3,node1)
x12=datnod(1,node2)
y12=datnod(2,node2)
z12=datnod(3,node2)
x13=datnod(1,node3)
y13=datnod(2,node3)
z13=datnod(3,node3)
isum=0
   do 39 kface=1,nfaces
   c nodes of the face number kface to check against
   ii=nbound(1,kface)
   np(1)=nconn(1,ii)
   39 continue
np(2)=nconn(2,i1)
i2=nbound(2,kface)
np(3)=nconn(1,i2)
np(4)=nconn(2,i2)
i3=nbound(3,kface)
np(5)=nconn(1,i3)
np(6)=nconn(2,i3)
c find the three unique points
node1=np(1)
node2=np(2)
do 613 ii=3,6
  npt=np(ii)
  if((npt.ne.np(1)).and.(npt.ne.np(2))) then
    c must be the third node
    node3=npt
go to 612
  endif
613 continue
612 continue
c node coordinates of the second face
x21=datnod(1,node1)
y21=datnod(2,node1)
z21=datnod(3,node1)
x22=datnod(1,node2)
y22=datnod(2,node2)
z22=datnod(3,node2)
x23=datnod(1,node3)
y23=datnod(2,node3)
z23=datnod(3,node3)
c see if the triangles overlap
  call facetck(x11,y11,z11,x12,y12,z12,x13,y13,
    & x31,y31,z31,x32,y32,z32,x33,y33,z33,iflag)
isum=isum+iflag
c if iflag.eq.0 there is no overlap so continue checking
  if(iflag.eq.0) go to 38
  c if iflag.eq.1 do not include the face being tested; no need to
  c check any more faces since this face is being discarded
  if(iflag.eq.1) go to 311
  c if iflag.eq.2 keep the face being tested and discard face number kface.
  if(iflag.eq.2) then
  c reduce the multiplicity factor of the discarded face edges by one
  c face number kface has edges i1,i2,i3
  nconn(3,i1)=nconn(3,i1)-1
  nconn(3,i2)=nconn(3,i2)-1
  nconn(3,i3)=nconn(3,i3)-1
  nbond(1,kface)=jedge
  nbond(2,kface)=itrak(jedge)
  nbond(3,kface)=itrak(kedge)
c increase the multiplicity factor of the new face edges by one.
nconn(3,k1)=nconn(3,k1)+1
nconn(3,k2)=nconn(3,k2)+1
nconn(3,k3)=nconn(3,k3)+1

c no need to check the remaining faces.
go to 311
endif
38 continue
39 continue

c made it all the way through -- add this edge
nfaces=nfaces+1
c put the face into nbound.
  nbound(1,nfaces)=iedge
  nbound(2,nfaces)=itrak(jedge)
  nbound(3,nfaces)=itrak(kedge)
c increment the multiplicity factor of the edges.
  nconn(3,iedge)=nconn(3,iedge)+1
  nconn(3,itrak(jedge))=nconn(3,itrak(jedge))+1
  nconn(3,itrak(kedge))=nconn(3,itrak(kedge))+1

c****************************************************************************************************************************************
  goto 311
endif
31 continue
30 continue
301 continue
311 continue
300 continue
100 continue

c at first nunknb=-nedges. this would be incremented by 3 for each
c face that was found. therefore nunknb=-nedges+3*nfaces.
  nunknb=3*nfaces-nedges
  return
end

c================================================================================================================================================
  subroutine facetck(x11,y11,z11,x12,y12,z12,x13,y13,z13,
    & x21,y21,z21,x22,y22,z22,x23,y23,z23,iflag)
c================================================================================================================================================

c check to see if two faces overlap based on the common
c surface area of circles inscribed in the two triangles
  iflag=0
c only test with triangles that lie in the same plane
c the condition is that the mags of the components of the
c cross product of the normals must be < eps
  eps=1.e-2
c notation: first number in a variable refers to face
c    second number refers to node or edge
c x,y,z components of the three edge vectors of triangle 1
  eg11x=x12-x11
  eg11y=y12-y11
  eg11z=z12-z11
\[ e_{12}x = x_{13} - x_{12} \]
\[ e_{12}y = y_{13} - y_{12} \]
\[ e_{12}z = z_{13} - z_{12} \]
\[ e_{13}x = x_{11} - x_{13} \]
\[ e_{13}y = y_{11} - y_{13} \]
\[ e_{13}z = z_{11} - z_{13} \]

**c edge lengths for face 1**
\[ e_{11}l = \sqrt{e_{11}x^2 + e_{11}y^2 + e_{11}z^2} \]
\[ e_{12}l = \sqrt{e_{12}x^2 + e_{12}y^2 + e_{12}z^2} \]
\[ e_{13}l = \sqrt{e_{13}x^2 + e_{13}y^2 + e_{13}z^2} \]

**c x,y,z components of the three edge vectors of triangle 2**
\[ e_{21}x = x_{22} - x_{21} \]
\[ e_{21}y = y_{22} - y_{21} \]
\[ e_{21}z = z_{22} - z_{21} \]
\[ e_{22}x = x_{23} - x_{22} \]
\[ e_{22}y = y_{23} - y_{22} \]
\[ e_{22}z = z_{23} - z_{22} \]
\[ e_{23}x = x_{21} - x_{23} \]
\[ e_{23}y = y_{21} - y_{23} \]
\[ e_{23}z = z_{21} - z_{23} \]

**c edge lengths for face 2**
\[ e_{21}l = \sqrt{e_{21}x^2 + e_{21}y^2 + e_{21}z^2} \]
\[ e_{22}l = \sqrt{e_{22}x^2 + e_{22}y^2 + e_{22}z^2} \]
\[ e_{23}l = \sqrt{e_{23}x^2 + e_{23}y^2 + e_{23}z^2} \]

**c unit vectors normal to each face**
\[ \text{call } AxB(e_{11}x, e_{11}y, e_{11}z, e_{12}x, e_{12}y, e_{12}z, v_{n1}x, v_{n1}y, v_{n1}z) \]
\[ \text{vmag1} = \sqrt{v_{n1}x^2 + v_{n1}y^2 + v_{n1}z^2} \]
\[ v_{n1}x = v_{n1}x / \text{vmag1} \]
\[ v_{n1}y = v_{n1}y / \text{vmag1} \]
\[ v_{n1}z = v_{n1}z / \text{vmag1} \]
\[ \text{call } AxB(e_{21}x, e_{21}y, e_{21}z, e_{22}x, e_{22}y, e_{22}z, v_{n2}x, v_{n2}y, v_{n2}z) \]
\[ \text{vmag2} = \sqrt{v_{n2}x^2 + v_{n2}y^2 + v_{n2}z^2} \]
\[ v_{n2}x = v_{n2}x / \text{vmag2} \]
\[ v_{n2}y = v_{n2}y / \text{vmag2} \]
\[ v_{n2}z = v_{n2}z / \text{vmag2} \]
\[ \text{call } AxB(v_{n1}x, v_{n1}y, v_{n1}z, v_{n2}x, v_{n2}y, v_{n2}z, v_{n}, v_{n}, v_{n}) \]

**c if \vert v_{n} \vert and \vert v_{n} \vert and \vert v_{n} \vert are sufficiently small then these two**

**c faces can be considered coincident**
\[ \text{if} \left( \text{abs}(v_{n}x) \lt \text{eps}. \text{and.} \text{abs}(v_{n}y) \lt \text{eps}. \text{and.} \text{abs}(v_{n}z) \lt \text{eps} \right) \]

**c areas of the two triangles**
\[ \text{call } AxB(x_{11}, y_{11}, z_{11}, x_{12}, y_{12}, z_{12}, a_{1}, b_{1}, c_{1}) \]
\[ \text{call } AxB(x_{12}, y_{12}, z_{12}, x_{13}, y_{13}, z_{13}, a_{2}, b_{2}, c_{2}) \]
\[ \text{call } AxB(x_{13}, y_{13}, z_{13}, x_{11}, y_{11}, z_{11}, a_{3}, b_{3}, c_{3}) \]
\[ \text{area1} = \sqrt{(a_{1} + a_{2} + a_{3})^2 + (b_{1} + b_{2} + b_{3})^2 + (c_{1} + c_{2} + c_{3})^2}/2. \]
\[ \text{call } AxB(x_{21}, y_{21}, z_{21}, x_{22}, y_{22}, z_{22}, a_{1}, b_{1}, c_{1}) \]
\[ \text{call } AxB(x_{22}, y_{22}, z_{22}, x_{23}, y_{23}, z_{23}, a_{2}, b_{2}, c_{2}) \]
\[ \text{call } AxB(x_{23}, y_{23}, z_{23}, x_{21}, y_{21}, z_{21}, a_{3}, b_{3}, c_{3}) \]
\[ \text{area2} = \sqrt{(a_{1} + a_{2} + a_{3})^2 + (b_{1} + b_{2} + b_{3})^2 + (c_{1} + c_{2} + c_{3})^2}/2. \]

42
c compute perimeters
  perim1=eg11m+eg12m+eg13m
  perim2=eg21m+eg22m+eg23m

c find position vectors to the centers of each inscribed circle
  c1x=(eg12m*x11+eg13m*x12+eg11m*x13)/perim1
  c1y=(eg12m*y11+eg13m*y12+eg11m*y13)/perim1
  c1z=(eg12m*z11+eg13m*z12+eg11m*z13)/perim1
  c2x=(eg22m*x21+eg23m*x22+eg21m*x23)/perim2
  c2y=(eg22m*y21+eg23m*y22+eg21m*y23)/perim2
  c2z=(eg22m*z21+eg23m*z22+eg21m*z23)/perim2

c distance between centers
  d12=sqrt((c1x-c2x)**2+(c1y-c2y)**2+(c1z-c2z)**2)

c radii of circles are areas/perimeters
  rad1=area1/perim1
  rad2=area2/perim2

c if d12 < rad1+rad2 then there is overlap
  if(d12.lt.(rad1+rad2)) then
    iflag=1
    if(rad2.gt.rad1) iflag=2
  endif
  return
end

subroutine AXB(a1,a2,a3,b1,b2,b3,c1,c2,c3)
c cross product of two vectors
  c1=a2*b3-a3*b2
  c2=a3*b1-a1*b3
  c3=a1*b2-a2*b1
return
end
APPENDIX B: ACAD-TO-PATCH TRANSLATOR CODE

The ACAD-to-PATCH translator code ATP is described in Section 5.1. It converts an ACAD "facet" file to a PATCH input file.

c program atp.f (version 3 -- allows subparts & uses face checking)
c "acad to patch" translator ENHANCED VERSION

c this program reads a file named "in.facet" then reformats the data
and writes it to the file named "out.patch" which can be read into
"build5.f" as a geometry file. characteristics of ACAD facet file:
>> a vertex table is given for each part (e.g., each face of a cube
is considered a part)
>> each part is assigned a name
>> node index is reinitialized for each part and nodes common to
several parts are present more than once (run "knit.f" on
"out.patch" to eliminate duplicate edges)
>> a vertex connection list is given for each part (face index and
its three vertex indices)
>> material parameters are included
>> multiple parts allowed, and each part can have several subparts
x,y,z are node coordinates (index is facet number)
datnod and nconn are same as in patch.f
dimension node(3,6000),nbond(3,6000),nv(6000),part(6000)
dimension datnod(3,6000),nconn(3,6000),istart(200)
dimension tmpdat(3,6000),icount(6000),indsum(6000)
dimension ivmin(6000),ivtx(6000,500),iskip(6000),indx(6000)
iinteger tmptod(3,6000)
character*80 title
c character*20 part,subpt
d distances less than eps are considered the same
eps=0.
c iverb=0 is verbose mode -- progress displayed
iverb=0
open(1, file='in.facet', status='old')
xmlns=1.e6
ymin=1.e6
zmin=1.e6
xmin=-1.e6
ymin=-1.e6
zmax=-1.e6

format(a80)
format(a20)
read(1,1) title
write(6,*) 'title:',title
read(1,*) nparts

44
c keep track of total number of faces and vertices
   nvtxl=0
   nftvl=0
   do 1000 npart=1, nparts
   read(1,2) part(npart)
   read(1,*), nmir
   if(nmir.ne.0) write(6,*) 'error: structure is mirrored'
   read(1,*) nv(npart)
   nverts=nv(npart)
   if(iverb.eq.0) then
     write(6,*) 'part number: ', npart
     write(6,*) ' name: ', part(npart)
     write(6,*) ' vertices: ',nv(npart)
   endif

c read node table for this subpart
   do 10 n=1,nverts
   read(1,*) xx,yy,zz
   nn=nvtxl+n
   tmdat(1,nn)=xx
   tmdat(2,nn)=yy
   tmdat(3,nn)=zz
   datnod(1,nn)=tmdat(1,nn)
   datnod(2,nn)=tmdat(2,nn)
   datnod(3,nn)=tmdat(3,nn)
   xmax=maxi(xmax,xx)
   ymax=maxi(ymax,yy)
   zmax=maxi(zmax,zz)
   xmin=mini(xmin,xx)
   ymin=mini(ymin,yy)
   zmin=mini(zmin,zz)
10   continue

c loop through subparts
   read(1,*), nspts
   if(iverb.eq.0) write(6,*)
   & 'number of subparts in current part: ', nspts
   do 15 kn=1,nspts
   read(1,2) subpt
   if(iverb.eq.0) then
     write(6,*) 'subpart number: ', kn
     write(6,*) ' name: ', subpt
   endif

c read: el type, no. faces, no. vertices, em2, vp, vn, ec

c restrictions: em2=0 (one-sided properties)
   vp=0 (vertex parameters)
   vn=0 (no normals present)
   ec=0 (no curvature lines)
   read(1,*), nsides,nfaces, nsiv, nem2, nvp, nvn, nec

c display if there are problems:
   if(nsides.ne.3)
& write(6,*) 'nontriangular facet encountered'
if(nsiv.ne.0) write(6,*) 'problem: nsiv is not zero'
if(nem2.ne.0) write(6,*) 'problem: EM2 is not zero'
if(nvp.ne.0) write(6,*) 'problem: VP is not zero'
if(nvn.ne.0) write(6,*) 'problem: VN is not zero'
if(nec.ne.0) write(6,*) 'problem: EC is not zero'
c if this is a part nodes=nverts; if this is a subpart nodes=nsiv
   do 20 n=1,nfaces
      nn=nn+nvtl
   c read vertex connection list
   read(1,*) nde1,nde2,nde3,ndum
   tmpnod(1,nn)=nde1+nvtl
   tmpnod(2,nn)=nde2+nvtl
   tmpnod(3,nn)=nde3+nvtl
   node(1,nn)=tmpnod(1,nn)
   node(2,nn)=tmpnod(2,nn)
   node(3,nn)=tmpnod(3,nn)
20   continue
   nvtl=nvtl+nfaces
15   continue
   nvte=nvtl+nv(npart)
1000 continue
   nverts=nvtl
   nfaces=nvtl
   write(6,*) 'total number of vertices, faces read=',nverts,nfaces
   c find duplicate vertices and save their indices in array iold.
   c icount(i) counts the number of times vertex i occurs
   do 75 iv1=1,nverts
      xi=tmpdat(1,iv1)
y1=tmpdat(2,iv1)
z1=tmpdat(3,iv1)
icount(iv1)=1
   c keep track of vertex number iv1 and its duplicates
   c ivtx(node number, occurrence number, duplicate index)
   ivtx(iv1,icount(iv1))=iv1
   do 73 iv2=1,nverts
      if(iv1.ne.iv2) then
         x2=tmpdat(1,iv2)
y2=tmpdat(2,iv2)
z2=tmpdat(3,iv2)
dx=abs(x1-x2)
dy=abs(y1-y2)
dz=abs(z1-z2)
         if((dx.lt.eps).and.(dy.lt.eps).and.(dz.lt.eps)) then
            icount(iv1)=icount(iv1)+1
            ivtx(iv1,icount(iv1))=iv2
         endif
      endif
73    continue
continue

for each vertex find the smallest index
    do 78 iv=1,nverts
        ivmin(iv)=nverts+1
    do 78 ii=1,icount(iv)
        ivmin(iv)=min(ivmin(iv),ivtx(iv,ii))
    continue

find duplicate vertices
    irem=0
    do 76 ii=1,nverts
        if (ivmin(ii).lt.iirem) then
            irem=irem+1
            iskip(irem)=ii
            endif
        continue

order the indices to be removed from lowest to highest
    idx=nverts
    do 85 ii=1,irem
        do 83 i2=1,irem-ii+1
            idx(ii)=min(idx,iskip(i2))
        continue
        idx=idx(ii)+1
    continue

if (iverb.eq.0) then
    do 190 ix=1,irem
        write(6,'(a10,2i10)') 'remove #:',ix,iskip(ix),idx(ix)
    endif

set all node indices to minimum value
    do 95 nn=1,nfaces
        do 95 is=1,3
            iv=tmpnod(is,nn)
            node(is,nn)=ivmin(iv)
        continue
        do 96 nn=1,nfaces
            do 96 is=1,3
                tmpnod(is,nn)=node(is,nn)
            continue
            do 100 ir=1,irem
                do 100 iv=indx(ir),nverts-ir
                    datnod(1,iv)=datnod(1,iv+1)
                    datnod(2,iv)=datnod(2,iv+1)
                    datnod(3,iv)=datnod(3,iv+1)
            continue
            nverts=nverts-irem

check each node of each face to see how many previous nodes have been removed (to determine the number of steps to decrement
c the index)
do 90 nn=1,nfaces
do 90 is=1,3
idx=tmpnod(is,nn)
c count vertices .lt. idx that have been removed
c nodes greater than iskip(ir) drop one; those less than iskip(ir) stay
iter=0
do 93 ir=1,irem
   c if(idx.eq.idx(ir)) go to 90
   do 93 ii=1,idx
       if(ii.eq.idx(ir)) iter=iter+1
93 continue
   node(is,nn)=tmpnod(is,nn)-iter
90 continue
   write(6,*) 'final number of vertices, faces=',nverts,nfaces

c rescale data if desired
c write(6,*) 'rescale data? (0=yes/1=no)'
   c read(5,*) ans
   ans=1
   if(ans.eq.0) then
      write(6,*) 'xmax,xmin=',xmax,xmin
      write(6,*) 'ymax,ymin=',ymax,ymin
      write(6,*) 'zmax,zmin=',zmax,zmin
      write(6,*) 'enter scale factor'
      read(5,*) fac
      do 50 n=1,nverts
         datnod(1,n)=fac*datnod(1,n)
         datnod(2,n)=fac*datnod(2,n)
         datnod(3,n)=fac*datnod(3,n)
50 continue
   endif

c generate edge connection list using "brute force" checking
   if(verb.eq.0)
 & write(6,*) 'start to generate edge connection list'
c for face n check pairs of vertices to see if they have been assigned
c an edge number. exception is n=1.
   nconn(1,1)=node(1,1)
nconn(2,1)=node(2,1)
nconn(1,2)=node(2,1)
nconn(2,2)=node(3,1)
nconn(1,3)=node(1,1)
nconn(2,3)=node(3,1)
edges=3
ncount=3
do 60 n=2,nfaces
   c check all edges to see if the three edges of the current face n
   c have already been assigned an index
   do 59 ns=1,3
      if(ns.eq.1) then
np1=node(1,n)
np2=node(2,n)
elseif(ns.eq.2) then
  np1=node(2,n)
np2=node(3,n)
else
  np1=node(3,n)
np2=node(1,n)
endif
if(ivert.eq.0) then
  write(6,'*') 'checking edge ',ns,' of face ',n
endif
do 65 ne=1,nedges
  c check edge ns of triangle n against edge ne
  mp1=nconn(1,ne)
  mp2=nconn(2,ne)
  if((mp1.eq.mp1).and.(mp2.eq.mp2)).or.
     &((mp2.eq.mp1).and.(mp1.eq.mp2)) then
  c edge is a duplicate and therefore no need to go further
     go to 66
  endif
  65 continue
  c made it all the way through -- must be new edge
  if(ivert.eq.0) then
    write(6,'*') 'new edge ',np1,np2
    write(6,'*') 'number of edges that have been defined is ',ncount
  endif
  ncount=ncount+1
  nconn(1,ncount)=np1
  nconn(2,ncount)=np2
  nedges=ncount
  66 continue
  59 continue
  60 continue
do 70 n=1,nedges
  70 nconn(3,n)=1
if(ivert.eq.0) write(6,'*') 'edge connection list generated'
c write reformatted data to file "out.patch"
open(2,file='out.patch')
write(2,'*') title
write(2,'*') nverts,nedges
do 151 n=1,nverts
  151 write(2,'*') n,datnod(1,n),datnod(2,n),datnod(3,n)
do 200 n=1,nedges
  200 write(2,'*') n,nconn(1,n),nconn(2,n)
c write new data in MATLAB files
open(12,file='xpts.m')
open(13,file='ypts.m')
open(14,file='zpts.m')
open(i5,file='end1.m')
open(i6,file='end2.m')

format(i5)
do 140 n=1,nverts
write(12,100) datnod(1,n)
write(13,100) datnod(2,n)
write(14,100) datnod(3,n)
140 continue
ormat(f15.4)
do 150 n=1,nedges
write(i5,101) nconn(1,n)
write(i6,101) nconn(2,n)
150 continue

c format(i5)
call geom(datnod,nconn,nedges,indsum,nbound,mxface,nfaces,
$ mxbnd,nunknb)
istart(i)=1
istart(2)=nfaces+1
call prntbnd(nconn,nbound,istart,1,nedges,nfaces,1)
c get body parameters
call bodpar(datnod,nconn,nbound,nverts,nedges,nfaces,nunknb)

998 continue
stop
end

=================================================================

subroutine prntbnd(nconn,nbound,istart,i,nedges,nfaces,nbodys)

=================================================================
c this subroutine prints the edges and the vertices of each face.
c input:
c nconn has the vertices and the multiplicity factor for each edge.
c nbound has the edges for each face.
c istart has the beginning faces for each body.
c i is the present body.
c nedges is the total number of edges.
c nfaces is the total number of faces.
c nbodys is the total number of bodys.

integer nconn(3,6000),nbound(3,6000),istart(200)
integer nverts(6000,3)
open(3,file='facelist')
open(9,file='facedat')
open(20,file='node1.m')
open(21,file='node2.m')
open(22,file='node3.m')
c loop through the faces of this body.
do 10 i10=istart(i),istart(i+1)-1
call facvtx(nconn,nedges,nbound(1,i10),nbound(2,i10),
>nbound(3,i10),nv1,nv2,nv3)
write(3,98)i10,nbound(1,i10),nbound(2,i10),nbound(3,i10),nv1,nv2
>,nv3
10 continue
c write face number and vertices to 'facedat'
  write(9,*)i10,nv1,nv2,nv3
  write(20,100) nv1
  write(21,100) nv2
  write(22,100) nv3
  nverts(i10,1)=nv1
  nverts(i10,2)=nv2
  nverts(i10,3)=nv3
10 continue
98 format(1x,'face',i5,' has edges',3i5,' with vertices',3i5)
100 format(i6)
return
end

=========================================
subroutine bodpar(datnod,nconn,nbound,nnodes,nedges,nfaces,numknb)
=========================================

c this subroutine computes the body parameters

c input:
c datnod(i,j) i=1,2,3 are the x,y,z coordinates of the jth node.
c nconn(3,j): edge j runs from node nconn(1,j) to node nconn(2,j)
c nconn(3,j) is the multiplicity of the jth edge.
c nbound(i,j): contains the ith edge of the jth face i=1,2,3.
c nnodes = the number of body nodes.
c nedges = the number of edges.
c nfaces = the number of faces.
c numknb = the number of body unknowns.
c output:
c aedgen = the average edge length(meters**2) including multiplicity.
c edgenx = the maximum edge length(meters).
c mxedge = the edge number of the edge with length edgenx.
c edgenn = the minimum edge length(meters).
c mnedge = the edge number of the edge with length edgenn.
c area = the surface area of the scatter(meters**2):for thin
      structures only one side is considered in the surface area.
c avera = the average area of the faces.
c mxarea = the number of the face with the maximum area(areaax).
c mxarea = the number of the face with the minimum area(areaam).
c ratio = the minimum height to base ratio over all faces.
c mmrtio = the face number that has a height to base ratio of 'ratio'.

dimension datnod(3,6000)
  integer nconn(3,6000),nbound(3,6000)
  common/params/avedgen,edgenx,mxedge,edgenn,mnedge,tarea,avarea,
                  $mxarea,mxarea,areaax,areaam,ratio,mmrtio
  common/mchval/valmax, valmin
  c save /params/
  c the following line is a statement function.
  size(x,y,z)=sqrt(x*x+y*y+z*z)
  c initialization.
  sedgl=0

51
edgemx=valmin
edgmn=valmax
    do 20 i=1,nedges
        mult=nconn(3,i)
        n1=nconn(1,i)
        n2=nconn(2,i)
        x=datnod(1,n2)-datnod(1,n1)
        y=datnod(2,n2)-datnod(2,n1)
        z=datnod(3,n2)-datnod(3,n1)
        edgl=size(x,y,z)
        sedgl=sedgl+mult*edgl
        if(edgl.gt.edgmx) then
            edgmx=edgl
            mxedge=i
        endif
        if(edgl.lt.edgmn) then
            edgmn=edgl
            mnedge=i
        endif
    20 continue
    avedge=sedgl/nunknb

    c compute tarea, avarea, mnarea, areamn, mxarea, areamax, ratio, and mrtio.
    ratio=valmax
    areamx=valmin
    areamn=valmax
    tarea=0.
    do 40 iface=1,nfaces
        is1=nbound(1,iface)
        is2=nbound(2,iface)
        is3=nbound(3,iface)
        call facvtx(nconn,nedges,is1,is2,is3,nv1,nv2,nv3)
        call vtcrd(datnod,nnodes,nv1,nv2,nv3,x1,x2,x3,y1,y2,y3,z1,z2,z3
          >
          x1mx3=x1-x3
          y1my3=y1-y3
          z1mz3=z1-z3
          x2mx3=x2-x3
          y2my3=y2-y3
          z2mz3=z2-z3
          x2mx1=x2-x1
          y2my1=y2-y1
          z2mz1=z2-z1
      c compute area of face by taking the cross product of two edge vectors.
          vx=y1my3*z2mz3-z1mz3*y2my3
          vy=z1mx3*x2mx3-x1mx3*z2mz3
          vz=x1mx3*y2my3-y1my3*x2mx3
          area=.5*size(vx,vy,vz)
    c compute the square of the lengths of each side.
        ris=x2mx3*x2mx3+y2my3*y2my3+z2mz3*z2mz3

52
r2s=x1mx3*x1mx3+y1my3*y1my3+z1mz3*z1mz3
r3s=x2mx1*x2mx1+y2my1*y2my1+z2mz1*z2mz1

c compute the height to base ratios.
area2=area+area
htb1=area2/ris
htb2=area2/r2s
htb3=area2/r3s
htbmin=min1(htb1,htb2,htb3)
tarea=tarea+area
if(area.gt.areamx)then
mxarea= iface
areamx= area
endif
if(area.lt.areamn)then
mnarea= iface
areamn= area
endif
if(htbmin.lt.ratio)then
mnratio= iface
ratio= htbmin
endif
40 continue
avarea=area/nfaces
write(3,110)
110 format(25x,'body parameter list'/)
write(3,111) nnodes,nedges,nfaces,nunknb
111 format(10x,'number of vertices=',i4,/10x,'number of edges=',i4,/10x
> 'number of faces=',i4,/10x,'number of edges including multiplici
> ty=',i4)
write(3,205)
205 format(25x,' modeling parameter list (meters) '/)
write(3,206) tarea
206 format(10x,'surface area of the scatterer=',e12.5,1x,'sq.meters')
write(3,209) avedge,mxedge,edgex,mnedge,edgemn
209 format(10x,'average edge length=',e12.5,1x,'meters',
'/>10x,'maximum edge length(edge no.,',i3,')=',e12.5,1x,'meters',
'/>10x,'minimum edge length(edge no.,',i3,')=',e12.5,1x,'meters')
write(3,210) avarea,mxarea,areamx,mnarea,areamn
210 format(10x,'average face area =',e12.5,1x,'sq.meters',/10x,
'/>10x,'maximum face area (face no.,',i4,1x,')=',e12.5,1x,'sq.meters',/
'/>10x,'minimum face area (face no.,',i4,1x,')=',e12.5,1x,'sq.meters')
write(3,211) mnratio,ratio
211 format(10x,'minimum face height to base ratio (face no.,'
'/>$i4,1x,')=',e11.5)
return
end

=================================================================================

subroutine facedg(nfaces,nbound,iface,iedg1,iedg2,iedg3)

=================================================================================

53
c this subroutine returns the edges of face number iface.
    integer nbound(3,6000)
    iedg1=nbound(1,iface)
    iedg2=nbound(2,iface)
    iedg3=nbound(3,iface)
    return
end

=================================================================
subroutine facvtx(nconn,edges,ie1,ie2,ie3,nv1,nv2,nv3)
=================================================================

c this subroutine returns the vertices of a given face

c nv1 is node opposite edge ie1.

c nv2 is node opposite edge ie2.

c nv3 is node opposite edge ie3.

    integer nconn(3,6000)

    c the node nv1 is the node that edges 2 and 3 have in common.
    c the node nv3 is the other node on edge 2.
    if(nconn(1,ie2).eq.nconn(1,ie3).or.nconn(1,ie2).eq.nconn(2,ie3))
      $then$
        nv1=nconn(1,ie2)
        nv3=nconn(2,ie2)
      $else$
        nv1=nconn(2,ie2)
        nv3=nconn(1,ie2)
      endif
    endif

c the node nv2 is the node that edges 1 and 3 have in common.
    if(nconn(1,ie1).eq.nconn(1,ie3).or.nconn(1,ie1).eq.nconn(2,ie3))
      $then$
        nv2=nconn(1,ie1)
      $else$
        nv2=nconn(2,ie1)
      endif
    return
end

=================================================================
subroutine vtxcrd(datnod,nnodes,n1,n2,n3,x1,x2,x3,y1,y2,y3,z1,z2,>
x3)
=================================================================

c this subroutine gets the coordinates of the vertices of a face

    dimension datnod(3,6000)
    x1=datnod(1,n1)
    y1=datnod(2,n1)
    z1=datnod(3,n1)
    x2=datnod(1,n2)
    y2=datnod(2,n2)
    z2=datnod(3,n2)
    x3=datnod(1,n3)
    y3=datnod(2,n3)
    z3=datnod(3,n3)
Subroutines GEOM, FACETCK, and AXB are used here. Listings appear in Appendix A.
APPENDIX C: PATCH-TO-ACAD TRANSLATOR CODE

As described in Section 5.2, there are two versions of the PATCH-to-ACAD translator: PTA and PTF. PTA translates the entire patch file geometry as a single part. PTF translates each facet as a single part. Only PTF is listed here. Therefore, in the case of PTF, the file length is much greater, but ACAD is able to manipulate each facet. Facet checking is used in subroutine GEOM.

```c
program ptf.f
  (similar to pta.f version 3 - uses INSCRIBED CIRCLE face check)
  
  "patch to facet" translator
  ***** differs from pta.f in that each triangle is ******
  an individual part  ******
  
  this program reads a file named "in.patch" then reformats the data
  and writes it to the file named "out.facet" which can be read into
  acad as a *.facet file
  x,y,z are node coordinates (index is facet number)
  datnod and nconn are same as in patch.f
    dimension node(3,6000),nbound(3,6000),np(6000),indsum(6000)
    dimension datnod(3,6000),nconn(3,6000),istart(6000)
    character*80 title
    character*19 dum1
    character*11 dum2
  verbose mode: iverb=0 displays progress
  iverb=0
  ******  read patch file  ******
    open(2,file='in.patch',status='old')
    read(2,11) title
  11 format(a80)
    read(2,*) nverts,nedges
    do 151 mn=1,nverts
    151  read(2,*) n,datnod(1,n),datnod(2,n),datnod(3,n)
    do 200 mn=1,nedges
    200  read(2,*) n,nconn(1,n),nconn(2,n)
  call array nbound with edges of each face
    call geom(datnod,nconn,nedges,indsum,nbound, nfaces,nunknb)
    do 201 nf=1,nfaces
    istart(nf)=nf
    istart(nf)=nf+1
    call printbd(nconn,nbound,istart,nf,nedges,nfaces,nfaces)
    201  continue
  get body parameters
    call bodpar(datnod,nconn,nbound,nverts,nedges,nfaces,nunknb)
  find the vertices of each face and put in array node
```

56
do 600 n=1,nf
if(iverb.eq.0) write(6,*) 'finding vertices of face ',n
i1=nbound(1,n)
np(1)=nconn(1,i1)
np(2)=nconn(2,i1)
i2=nbound(2,n)
np(3)=nconn(1,i2)
np(4)=nconn(2,i2)
i3=nbound(3,n)
np(5)=nconn(1,i3)
np(6)=nconn(2,i3)
c find the three unique points
node(1,n)=np(1)
node(2,n)=np(2)
ieg=3
do 603 ii=3,6
npt=np(ii)
if((npt.ne.ny(1)).and.(npt.ne.ny(2))) then
c must be the third node
node(3,n)=npt
goto 602
endif
603 continue
602 continue
if(iverb.eq.0) write(6,*) 'vertices are: ',(node(jj,n),jj=1,3)
600 continue
c ************** write facet file **********************
title='FACET FILE V3.0  SG4D '
open(1,file='outfacet')
c ignoring material parameters
write(1,1) title
c number of parts is the number of facets
write(1,2) nf
do 20 n=1,nf
write(1,13) ' FFace ',n
13 format(a7,i4)
write(1,4) 0
c each facet has 3 vertices
write(1,4) 3
do 40 m=1,3
write(1,8) datnodd(1,node(m,n)),datnodd(2,node(m,n)),
 & datnodd(3,node(m,n))
40 continue
write(1,2) 1
write(1,*) 'Tri Sheet 0'
write(1,7) 3,1,0,0,0,0,0
write(1,9) 1,2,3
20 continue
1 format(a28)
c write new data in MATLAB files
    open(12,file='xpts.m')
    open(13,file='ypts.m')
    open(14,file='zpts.m')
    open(15,file='end1.m')
    open(16,file='end2.m')

90 format(i5)
   do 140 n=1,nverts
    write(12,100) datnod(1,n)
    write(13,100) datnod(2,n)
    write(14,100) datnod(3,n)
   140 continue
100 format(f15.4)
   do 150 n=1,nedges
    write(15,101) nconn(1,n)
    write(16,101) nconn(2,n)
   150 continue
101 format(i5)
998 continue
stop
end

subroutine prntbd(nconn,nbound,istart,i,nedges,nfaces,nbodys)

This subroutine prints the edges and the vertices of each face.

Input:
- nconn has the vertices and the multiplicity factor for each edge.
- nbound has the edges for each face.
- istart has the beginning faces for each body.
- i is the present body.
- nedges is the total number of edges.
- nfaces is the total number of faces.
- nbodys is the total number of bodys.

integer nconn(3,7000),nbond(3,7000),istart(7000)
integer nvverts(7000,3)
open(3,file='facelist')
open(9,file='facedat')
open(20,file='node1.m')
open(21,file='node2.m')
open(22,file='node3.m')

C loop through the faces of this body.
do 10 i10=istart(i),istart(i+1)-1
   call facvtx(nconn,nedges,nbound(1,i10),nbound(2,i10),
>          nbound(3,i10),nv1,nv2,nv3)
   write(3,98)i10,nbound(1,i10),nbound(2,i10),nbound(3,i10),nv1,nv2
   >,nv3
   c write face number and vertices to 'facedat'
   write(9,*),i10,nv1,nv2,nv3
   write(20,100) nv1
   write(21,100) nv2
   write(22,100) nv3
   nverts(i10,1)=nv1
   nverts(i10,2)=nv2
   nverts(i10,3)=nv3
   10 continue
98  format(ix,'face',i5,' has edges',3i5,' with vertices',3i5)
100 format(i8)
   return
end

subroutine bodpar(datnod,nconn,nbound,nnodes,nedges,nfaces,nunknb)

this subroutine computes the body parameters
input:
  datnod(i,j) i=1,2,3 are the x,y,z coordinates of the jth node.
  nconn(3,j): edge j runs from node nconn(1,j) to node nconn(2,j)
  nconn(3,j) is the multiplicity of the jth edge.
  nbound(i,j): contains the ith edge of the jth face i=1,2,3.
  nnodes = the number of body nodes.
  nnedges = the number of edges.
  nnfaces = the number of faces.
  nunknb = the number of body unknowns.
output:
  avedge = the average edge length(meters**2) including multiplicity.
  edgemx = the maximum edge length(meters).
  mnxedge = the edge number of the edge with length edgemx.
  edgmx = the minimum edge length(meters).
  mnedge = the edge number of the edge with length edgmx.
  tarea = the surface area of the scatter(meters**2):for thin
  structures only one side is considered in the surface area.
  avara = the average area of the faces.
  mxarea = the number of the face with the maximum area(areamx).
  mnarea = the number of the face with the minimum area(areamn).
  ratio = the minimum height to base ratio over all faces.
  mnrtio = the face number that has a height to base ratio of 'ratio'.

dimension datnod(3,6000)
integer nconn(3,6000),nbound(3,6000)
common/params/avedge,edgemx,mnxedge,edgmx,mnedge,tarea,avarea,
$xarea,mxarea,areamx,areamn,ratio,mnrtio
common/mchval/valmax,valmin

59
c save /params/
c the following line is a statement function.
    size(x,y,z)=sqrt(x*x+y*y+z*z)
c initialization.
    sedgl=0
    edgemx=valmin
    edgmn=valmax
    do 20 ie=1,nedges
        mult=nconn(3,ie)
        ni=nconn(1,ie)
        n2=nconn(2,ie)
        x=datnod(1,n2)-datnod(1,ni)
        y=datnod(2,n2)-datnod(2,ni)
        z=datnod(3,n2)-datnod(3,ni)
        edgl=size(x,y,z)
        sedgl=sedgl+mult*edgl
    if(edgl.gt.edgemx) then
        edgemx=edgl
        mxxedge=ie
    endif
    if(edgl.lt.edgmn) then
        edgmn=edgl
        mnedge=ie
    endif
 20  continue
    avedge=sedgl/nunknb

c compute tarea,avarea,nnarea,areaam,mxarea,areaamx,ratio,and mmrtio.
    ratio=valmax
    areamx=valmin
    areamn=valmax
    tarea=0.
    do 40 iface=1,nfaces
        is1=nbound(1,iface)
        is2=nbound(2,iface)
        is3=nbound(3,iface)
        call facvtx(nconn,nedges,is1,is2,is3,nv1,nv2,nv3)
        call vtxcrd(datnod,nmodes,nv1,nv2,nv3,x1,x2,x3,y1,y2,y3,z1,z2,z3)
    >
        x1mx3=x1-x3
        y1my3=y1-y3
        z1mz3=z1-z3
        x2mx3=x2-x3
        y2my3=y2-y3
        z2mz3=z2-z3
        x2mx1=x2-x1
        y2my1=y2-y1
        z2mz1=z2-z1

c compute area of face by taking the cross product of two edge vectors.
    vx=y1my3*z2mz3-z1mz3*y2my3
\[ vy = z_{m3} \times x_{2m3} - x_{1m3} \times z_{2m3} \]
\[ vz = x_{1m3} \times y_{2m3} - y_{1m3} \times x_{2m3} \]
\[ area = 0.5 \times \text{size}(vx, vy, vz) \]

c compute the square of the lengths of each side.
\[ ris = x_{2m3} \times x_{2m3} + y_{2m3} \times y_{2m3} + z_{2m3} \times z_{2m3} \]
\[ r2s = x_{1m3} \times x_{1m3} + y_{1m3} \times y_{1m3} + z_{1m3} \times z_{1m3} \]
\[ r3s = x_{2m1} \times x_{2m1} + y_{2m1} \times y_{2m1} + z_{2m1} \times z_{2m1} \]

c compute the height to base ratios.
\[ area2 = area + area \]
\[ htb1 = area2 / ris \]
\[ htb2 = area2 / r2s \]
\[ htb3 = area2 / r3s \]
\[ htbmin = \text{min}(htb1, htb2, htb3) \]
\[ tare2 = area + area \]
\[ if(area > area x) then \]
\[ mxarea = iface \]
\[ area x = area \]
\[ endif \]
\[ if(area .lt. area m) then \]
\[ mnarea = iface \]
\[ area m = area \]
\[ endif \]
\[ if(htbmin .lt. ratio) then \]
\[ mnrtio = iface \]
\[ ratio = htbmin \]
\[ endif \]

40 continue

avarea = tarea / nfaces
write(3, 110)

110 format(25x, 'body parameter list'/)
write(3, 111) nmod, nedges, nfaces, nunknb

111 format(10x, 'number of vertices=', i4, /10x, 'number of edges=', i4, /10
> x, 'number of faces=', i4, /10x, 'number of edges including multipli
ci'

> ty=', i4)
write(3, 205)

205 format(25x, 'modeling parameter list (meters)'/)
write(3, 206) tare2

206 format(10x, 'surface area of the scatterer=', e12.5, i1, 'sq.meters'/)
write(3, 209) averge, mxedge, edgenx, medge, edgem

209 format(10x, 'average edge length=', e12.5, i1, 'meters',
$/10x, 'maximum edge length(edge no.=', i3, ')=', e12.5, i1, 'meters',
$/10x, 'minimum edge length(edge no.=', i3, ')=', e12.5, i1, 'meters' )
write(3, 210) average, mxarea, area x, mnarea, area m

210 format(10x, 'average face area =', e12.5, i1, 'sq.meters', /10x,
$/'maximum face area (face no.=', i4, i1, ')=' , e12.5, i1, 'sq.meters', /
$/10x, 'minimum face area (face no.=', i4, i1, ')=' , e12.5, i1, 'sq.meters')
write(3, 211) mnrtio, ratio

211 format(10x, 'minimum face height to base ratio (face no.=',
$/i4, i1, ')=' , e11.5)
return
end

subroutine facedg(nfaces,nbound,iface,iedg1,iedg2,iedg3)
c this subroutine returns the edges of face number iface.
integer nbound(3,7000)
iedg1=nbound(1,iface)
iedg2=nbound(2,iface)
iedg3=nbound(3,iface)
return
end

subroutine facvtx(nconn,nedges,ie1,ie2,ie3,nv1,nv2,nv3)
c this subroutine returns the vertices of a given face
c nv1 is node opposite edge ie1.
c nv2 is node opposite edge ie2.
c nv3 is node opposite edge ie3.
integer nconn(3,7000)
c the node nv1 is the node that edges 2 and 3 have in common.
c the node nv3 is the other node on edge 2.
  if(nconn(1,ie2).eq.nconn(1,ie3).or.nconn(1,ie2).eq.nconn(2,ie3))
$then
    nv1=nconn(1,ie2)
    nv3=nconn(2,ie2)
  else
    nv1=nconn(2,ie2)
    nv3=nconn(1,ie2)
end
  c the node nv2 is the node that edges 1 and 3 have in common.
  if(nconn(1,ie1).eq.nconn(1,ie3).or.nconn(1,ie1).eq.nconn(2,ie3))
$then
    nv2=nconn(1,ie1)
  else
    nv2=nconn(2,ie1)
end
return
end

subroutine vtxcrd(datnod,nnodes,n1,n2,n3,x1,x2,x3,y1,y2,y3,z1,z2,z3)
c this subroutine gets the coordinates of the vertices of a face
dimension datnod(3,6000)
x1=datnod(1,n1)
y1=datnod(2,n1)
z1=datnod(3,n1)
x2=datnod(1,n2)
\begin{verbatim}
y2=datnod(2,n2)
z2=datnod(3,n2)
x3=datnod(1,n3)
y3=datnod(2,n3)
z3=datnod(3,n3)
return
end
\end{verbatim}

Subroutines GEOM, FACETCK, and AXB are used here. Listings appear in Appendix A.
APPENDIX D: PATCH-TO-NEC TRANSLATOR CODE

The PATCH-to-NEC translator allows the user to convert a PATCH input file into a format that can be read by NEC. CAUTION: The correspondence is only approximate and some errors may result as noted in Section 5.3

c program ptn.f
c "patch to NEC" translator
c this program reads a file named "in.patch" then reformats the data
c and writes it to the file named "out.nec" which can be read by NEC.
c x,y,z are node coordinates (index is facet number)
c datnod and nconn are same as in patch.f
c if iflag=0 edges in the xy plane (i.e., z < eps) are deleted
(omitted in the NEC GW list)
dimension datnod(3,8000),nconn(3,8000)
character*80 title
iflag=0
eps=1.e-6

c ****************** read patch file ********************
open(2,file='in.patch',status='old')
read(2,12) title
12 format(a80)
read(2,*) nverts,nedges
do 151 nn=1,nverts
151 read(2,*) n,datnod(1,n),datnod(2,n),datnod(3,n)
do 200 mn=1,nedges
200 read(2,*) n,nconn(1,n),nconn(2,n)
c ****************** write NEC file ******************
c patch edge index becomes tag number
c radius of each wire is set to 1/10 of its length
open(1,file='out.nec')
do 10 ne=1,nedges
c coordinates of first end
 n1=nconn(1,ne)
x1=datnod(1,n1)
y1=datnod(2,n1)
z1=datnod(3,n1)
c coordinates of second end
 n2=nconn(2,ne)
x2=datnod(1,n2)
y2=datnod(2,n2)
z2=datnod(3,n2)
c if iflag=0 and this segment lies in the xy plane omit it from
c the GW list. Note that the edge list is not compacted, only
c they are omitted in the write
if(iflag.eq.0) then
if((z1.lt.ep).and.(z2.lt.ep)) go to 20
endif
seg1=sqrt((x2-x1)**2+(y2-y1)**2+(z2-z1)**2)

C set radius for this segment
r0=seg1/10.
if(r0.gt.0.2) r0=0.2
if(r0.lt.0.02) r0=0.02
write(1,300) 'GW',ne,1,x1,y1,z1,x2,y2,z2,r0
20   continue
10   continue
300  format(a2,ix,2(i4,1x),6(f7.2,2x),f5.2)
    write(1,400) 'GE',1,2
    write(1,400) 'GN',1
    write(1,400) 'EN'
400  format(a2,2x,ii,2x,ii)
    stop
end
APPENDIX E: PATCH INPUT CHECKING CODE (KNIT)

KNIT checks a PATCH input file for duplicate edges and nodes.

c program knit.f  (final version 4: 12/10/95)
c
c removes duplicate nodes in a inpatch file & NOMAT
finds lines that cross or are parallel
(an edge that has the same node at both ends. this
c allows triangles to be built from quadrilaterals.)
c edge and node tags do not have to be ordered in 'inknit'
c edge numbers out will be different than edge numbers in
c datnod is the same as in patch.f
dimension new(5000),new(5000),datnod(3,5000),nnord(5000)
dimension tmpdat(3,5000),icount(5000),neord(5000)
dimension ivmin(5000),ivtx(5000,2000),iskip(5000),indx(5000)
dimension nedge(5000),node(5000),nflag(5000),mflag(5000)
type nmod(5000),end2(5000),endi(5000)
character*80 title
character*8 fin, fout
data nflag/5000*0/,mflag/5000*0/,iscl/1/,iverb/0/
c distances less than eps are considered the same
eps=1.e-3
c iverb=0: verbose mode -- progress displayed
c iscl=0: rescale data
  iverb=1
  xmin=1.e6
  ymin=1.e6
  zmin=1.e6
  xmax=-1.e6
  ymax=-1.e6
  zmax=-1.e6

****************************************************************************************
c read the inpatch file named inknit
****************************************************************************************
write(6,*)
& 'enter file name to remove duplicate edges and nodes:'
read(5,1) fin
1 format(a8)
write(6,*)
& 'enter output file name (<9 char & different from input name)'
read(5,1) fout
open(1,file=fin,status='old')
open(2,file=fout)
read(1,2) title
2 format(a80)
read(1,*) nverts,nedges
write(6,*) 'nverts,nedges=',nverts,nedges

66
nodes=nverts
do 10 n=1,nverts
read(1,*) node(n),xx,yy,zz
xmax=amax1(xmax,xx)
ymax=amax1(ymax,yy)
zm=amax1(zmax,zz)
xmin=amin1(xmin,xx)
ymin=amin1(ymin,yy)
zmin=amin1(zmin,zz)
tmpdat(1,n)=xx
tmpdat(2,n)=yy
tmpdat(3,n)=zz
datnod(1,n)=tmpdat(1,n)
datnod(2,n)=tmpdat(2,n)
datnod(3,n)=tmpdat(3,n)
tmpnod(n)=node(n)
c if the node has been read change the flag from 0 to 1
  if(nflag(node(n)).eq.0) nflag(node(n))=1
10 continue
do 20 n=1,nedges
read(1,*) nedge(n),end1(n),end2(n)
new1(n)=end1(n)
ewline2(n)=end2(n)
c if the edge has been read change the flag from 0 to 1
  if(nflag(nedge(n)).eq.0) nflag(nedge(n))=1
20 continue
c check to see if any nodes have not been read
do 30 n=1,nverts
  if(nflag(n).eq.0) then
    write(6,*),'node index skipped: ',n
  endif
30 continue
c check to see if any edges have not been read
do 35 n=1,nedges
  if(nflag(n).eq.0) then
    write(6,*),'edge index skipped: ',n
  endif
35 continue
write(6,*),'finished reading file inknit'

*****************************************************************************
c order node numbers from lowest to highest in case they have not
c been defined this way
do 573 n1=1,nodes
do 570 n2=1,nodes
c find edge n1 and save
  if(node(n2).eq.n1) then
    nnord(n1)=n1
    datnod(1,n1)=tmpdat(1,n2)
datnod(2,n1)=tmpdat(2,n2)
datnod(3,n1)=tmpdat(3,n2)
go to 572
endif
570 continue
572 continue
573 continue
do 575 n=1,nodes
    tmpdat(1,n)=datnod(1,n)
    tmpdat(2,n)=datnod(2,n)
    tmpdat(3,n)=datnod(3,n)
    node(n)=nnord(n)
575 continue
write(6,*) 'nodes re-ordered from low to high'
c find duplicate vertices and save their indices
c icount(i) counts the number of times vertex i occurs
do 75 iv1=1,nverts
    xi=tmpdat(1,iv1)
    yi=tmpdat(2,iv1)
    zi=tmpdat(3,iv1)
    icount(iv1)=1
75 continue
icount(iv1)
ivtx(iv1,icount(iv1))=iv1
do 73 iv2=1,nverts
    if(iv1.ne.iv2) then
        x2=tmpdat(1,iv2)
        y2=tmpdat(2,iv2)
        z2=tmpdat(3,iv2)
        dx=abs(xi-x2)
        dy=abs(yi-y2)
        dz=abs(zi-z2)
        if((dx.lt.eps).and.(dy.lt.eps).and.(dz.lt.eps)) then
            icount(iv1)=icount(iv1)+1
            ivtx(iv1,icount(iv1))=iv2
        endif
    endif
73 continue
75 continue
if(iverb.eq.0) write(6,*) 'array ivtx filled'
c for each vertex find the smallest index
do 78 iv=1,nverts
    ivmin(iv)=nverts+1
    do 78 ii=1,icount(iv)
        ivmin(iv)=min(ivmin(iv),ivtx(iv,ii))
    78 continue
if(iverb.eq.0) write(6,*) 'found smallest index'
c find duplicate vertices
irem=0
do 76 ii=1,nverts
c if minimum index is .lt. ii this is a duplicate
   if(ivmin(ii).lt.ii) then
      irem=irem+1
      iskip(irem)=ii
   endif
76 continue
   if(iverb.eq.0) write(6,*) 'duplicate vertices tagged'
    c order the indices to be removed from lowest to highest
    if(iverb.eq.0) write(6,*) 'start to re-order indices'
    idx=nverts
    do 85 ii=1,irem
       do 83 i2=1,irem-ii+1
          indx(ii)=min(indx(ii),iskip(i2))
    83       continue
    85       continue
    if(iverb.eq.0) then
       write(6,*) 'skip','','ordered'
    do 190 ix=1,irem
    190      write(6,*) iskip(ix),' ',indx(ix)
   endif

******************************************************************************
c fill array tmpnod with original node values
   do 196 nn=1,nverts
      tmpnod(nn)=node(nn)
196 continue
c set all node indices to minimum value
   do 95 nn=1,nverts
      iv=tmpnod(nn)
      node(nn)=ivmin(iv)
95 continue
c refill tmpnod with minimum node values
   do 96 nn=1,nverts
c set all endpoints to minimum node values
   do 91 kk=1,nedges
      if(end1(kk).eq.tmpnod(nn)) new1(kk)=node(nn)
      if(end2(kk).eq.tmpnod(nn)) new2(kk)=node(nn)
91 continue
96 continue
do 192 kk=1,nedges
   end1(kk)=new1(kk)
   end2(kk)=new2(kk)
192 continue
c remove duplicate nodes and shift remaining nodes down one for
c each previous node removed
   do 79 ir=1,irem
      do 79 iv=indx(ir),nverts-ir
         datnod(1,iv)=datnod(1,iv+1)
         datnod(2,iv)=datnod(2,iv+1)
79    continue
datnod(3, iv) = datnod(3, iv+1)

continue

c check each node to see how many previous nodes have been removed
c (to determine the number of steps to decrement the index)
do 90 n=1, nverts
   idx = tmpnod(n)

c count vertices .lt. idx that have been removed
c nodes greater than iskip(ir) drop one; those less than iskip(ir) stay
   iter = 0
   do 93 ir = 1, irem
      if (idx.eq.indx(ir)) go to 90
      do 93 ii = 1, idx
         if (ii.eq.indx(ir)) iter = iter + 1
   93 continue
   it = it + 1
   node(it) = tmpnod(nn) - iter

   c find all edges with this node and also shift them down
   c note end1 & end2 have original node numbers; new1 & new2 have new
   c node numbers

      do 179 kk = 1, nedges
         if (end1(kk).eq.tmpnod(nn)) new1(kk) = node(it)
         if (end2(kk).eq.tmpnod(nn)) new2(kk) = node(it)
   179 continue
   nverts = nverts - irem
write(6, *) 'final number of vertices =', nverts
   do 28 i = 1, nedges
      end1(i) = new1(i)
   28 end2(i) = new2(i)

******************************************************************************
c rescale data if desired
******************************************************************************

   if (iscl.eq.0) then
      write(6, *) 'rescale data? (0=yes/1=no)'
      read(5, *) ans
   if (ans.eq.0) then
      write(6, *) 'xmax,xmin=', xmax, xmin
      write(6, *) 'ymax,ymin=', ymax, ymin
      write(6, *) 'zmax,zmin=', zmax, zmin
      write(6, *) 'enter scale factor'
      read(5, *) fac
      do 50 n = 1, nverts
         datnod(1, n) = fac * datnod(1, n)
         datnod(2, n) = fac * datnod(2, n)
         datnod(3, n) = fac * datnod(3, n)
   50 continue
   endif
endif

******************************************************************************
c remove duplicate edges and use new node numbers

***************
c order edge numbers from lowest to highest in case they have not
c been defined this way
  do 473 n1=1,nedges
    do 470 n2=1,nedges
       c find edge n1 and save
          if(nedge(n2).eq.n1) then
            neord(n1)=n1
            new1(n1)=endi(n2)
            new2(n1)=end2(n2)
            go to 472
          endif
        470       continue
      472      continue
    473      continue
    do 471 nn=1,nedges
      if(iverb.eq.0) then
        write(6,'(a01)') 'nn,nedge,neord=',nn,nedge(nn),neord(nn)
      endif
      nedge(nn)=neord(nn)
      endi(nn)=new1(nn)
      end2(nn)=new2(nn)
    471      continue
    write(6,'(a01)') 'edges re-ordered from low to high'

***************

inon=0
  do 300 n=1,nedges
    if(endi(n).eq.end2(n)) then
      inon=innon+1
    endif
  300 continue
  nnone=n,nedges+n=1-inon
  endi(nn)=endi(nn+1)
  end2(nn)=end2(nn+1)
  new1(nn)=new1(nn+1)
  new2(nn)=new2(nn+1)
  nedge(nn)=nn
  neord(nn)=nn

310 continue
  endif

300 continue
nedges=nedges-inon

write(6,'(a01)') 'number of nonedges found was ',inon
  if(iverb.eq.0) then
    write(6,'(a01)') 'start to generate edge connection list'
  endif

  c search for duplicate edges
  c for each edge (n=1,nedges) check the vertices to see if they have
  c been previously assigned an edge number. save the unique set of
c endpoints in end1 and end2. length will be ncount.
  do 175 ie1=1,nedges
    icount(ie1)=1
    np1=end1(ie1)
    np2=end2(ie1)
  c keep track of vertex number iv1 and its duplicates
  c ivtx(node number, occurrence number, duplicate index)
    ivtx(ie1,icount(ie1))=ie1
  do 173 ie2=1,nedges
    if(ie1.ne.ie2) then
      mp1=end1(ie2)
      mp2=end2(ie2)
      if((np1.eq.mp1).and.(np2.eq.mp2)).or.
         ((np2.eq.mp1).and.(np1.eq.mp2))) then
        c .or.(np1.eq.mp2)) then
          icount(ie1)=icount(ie1)+1
          ivtx(ie1,icount(ie1))=ie2
        endif
      endif
    173 continue
  175 continue
  c for each edge find the smallest index
  do 178 ie=1,nedges
    ivmin(ie)=nedges+1
  do 178 ii=1,icount(ie)
    ivmin(ie)=min(ivmin(ie),ivtx(ie,ii))
  178 continue
  c find duplicate edges
    irem=0
  do 176 ii=1,nedges
    c if minimum index is .lt. ii this is a duplicate
      if(ivmin(ii).lt.ii) then
        irem=irem+1
        iskip(irem)=ii
      endif
    176 continue
  c order the edges to be removed from lowest to highest
  write(6,*)'number of duplicate edges=',irem
  if(iverb.eq.0) write(6,*)'re-ordering edges'
    idx=nedges
  do 185 ii=1,irem
    do 183 i2=1,irem-ii+1
      idx(ii)=min(idx,iskip(i2))
    183 continue
    idx=idx(ii)+1
  185 continue
  if(iverb.eq.0) then
    write(6,*)'skip',' ordered'
  do 191 ix=1,irem
write(6,*) iskip(ix),',indx(ix)
endif

remove duplicate edges and shift remaining edges down one for

c each previous edge removed
do 279 ir=1,irem
    do 279 ie=indx(ir),nedges-ir
        new1(ie)=new1(ie+1)
        new2(ie)=new2(ie+1)
c    neord(ie)=neord(ie+1)
279 continue

nedges=nedges-irem
write(6,*) 'final number of edges =',nedges
if(iverb.eq.0) write(6,*) 'edge connection list generated'

<<<< THIS PORTION OF THE CODE HAS NOT BEEN VALIDATED >>>><

c find lines that cross or overlap

c write(6,*)
& '***************
c do 909 na=1,nedges
    ne=nedge(na)
c    ni=end1(ne)
c    n2=end2(ne)
c    xi=datnod(1,ni)
c    y1=datnod(2,ni)
c    zi=datnod(3,ni)
c    x2=datnod(1,n2)
c    y2=datnod(2,n2)
c    z2=datnod(3,n2)
c do 908 mb=1,nedges
    me=nedge(mb)
c dont check if same edge
    if(me.le.ne) go to 907
    m1=end1(me)
c    m2=end2(me)
c at this point it has been assumed that all duplicate edges have
been removed. don't want to consider lines that have a common end point
    if((ni.eq.m1).or.(n2.eq.m2)).or.
c & (n1.eq.m2).or.(n2.eq.m1)) go to 907
    x3=datnod(1,m1)
c    y3=datnod(2,m1)
c    z3=datnod(3,m1)
c    x4=datnod(1,m2)
c    y4=datnod(2,m2)
c    z4=datnod(3,m2)
c write(6,*) 'calling intersect for edges ',me,ne
c call intersect(x1,y1,z1,x2,y2,z2,x3,y3,z3,
c & x4,y4,z4,iflag,ipar)
c if(ipar.eq.0)
c & write(6,*) 'WARNING - edges overlap: ',me,ne
c if(iflag.eq.0)
907 continue
908 continue
909 continue
write(6,*)
& '******************************************************
**write reformat data to file
******************************************************************************
open(2,file=fout)
write(2,2) title
write(2,*) nverts,nedges
do 151 n=1,nverts
151 write(2,*) node(n),datnod(1,n),datnod(2,n),datnod(3,n)
do 200 n=1,nedges
  nedge(n)=neord(n)
200 write(2,*) nedge(n),new1(n),new2(n)
c write new data in MATLAB files
open(12,file='xpts.m')
open(13,file='ypts.m')
open(14,file='zpts.m')
open(15,file='end1.m')
open(16,file='end2.m')
98 format(i5)
do 140 n=1,nverts
  write(12,100) datnod(1,n)
  write(13,100) datnod(2,n)
  write(14,100) datnod(3,n)
140 continue
100 format(f15.4)
do 150 n=1,nedges
  write(15,101) new1(n)
  write(16,101) new2(n)
150 continue
101 format(i5)
998 continue
stop
end
subroutine intersect(x1,y1,z1,x2,y2,z2,x3,y3,z3,
& x4,y4,z4,iflag,ipar)
c subroutine to find the intersection of two lines
c endpoints of line 1: (x1,y1,z1) and (x2,y2,z2)
c endpoints of line 2: (x3,y3,z3) and (x4,y4,z4)
c if there is an intersection iflag=0
c ipar=0 denotes overlapping lines
tol=1.e-5
iflag=1
ipar=1
a1=x2-x1
b1=y2-y1
c1=z2-z1
a2=x4-x3
b2=y4-y3
c2=z4-z3
v1=b1*c2-b2*c1
v2=c1*a2-c2*a1
v3=a2*b1-a1*b2
d1=x3-x1
d2=y3-y1
d3=z3-z1
rdotv2=d1*a2+d2*b2+d3*c2
rdotv1=d1*a1+d2*b1+d3*c1
v1dotv2=a1*a2+b1*b2+c1*c2
v2dotv2=a2*a2+b2*b2+c2*c2
v1v2sq=v1**2+v2**2+v3**2
if(v1v2sq.lt.tol) v1v2sq=0.
c check to see if the lines are parallel; set ipar=0 if cross
product is zero
if(v1v2sq.eq.0.) then
  c if these lines are parallel see if they overlap. note it is
  c assumed that this subroutine is not called for the same edge
  c ******* case 1: none of the coeffs are zero ******
  if((abs(a1).gt.tol).and.(abs(b1).gt.tol).and.
    & (abs(c1).gt.tol)) then
c    write(6,*) 'case1'
p=(x3-x1)/a1
q=(y3-y1)/b1
r=(z3-z1)/c1
    & (abs(p-r).gt.tol)) ipar=0
p=(x4-x1)/a1
q=(y4-y1)/b1
r=(z4-z1)/c1
    & (abs(p-r).gt.tol)) ipar=0
endif
  c ******* case 2: one of the coeffs are zero ******
c if a1=0 compare y and z
  if((abs(a1).lt.tol).and.(abs(b1).gt.tol).and.
    & (abs(c1).gt.tol)) then
c    write(6,*) 'case2a'
q=(y3-y1)/b1
r=(z3-z1)/c1
  if(abs(r-q).gt.tol) ipar=0
q=(y4-y1)/b1
r=(z4-z1)/c1
  if(abs(r-q).gt.tol) ipar=0
endif
endif

c if b1=0 compare x and z
   if((abs(b1).lt.tol).and.(abs(a1).gt.tol).and.
      (abs(c1).gt.tol)) then
      write(6,*) 'case2b'
p=(x3-x1)/a1
r=(z3-z1)/c1
if(abs(p-r).gt.tol) ipar=0
p=(x4-x1)/a1
r=(z4-z1)/c1
if(abs(p-r).gt.tol) ipar=0
endif

c if c1=0 compare x and y
   if((abs(c1).lt.tol).and.(abs(b1).gt.tol).and.
      (abs(a1).gt.tol)) then
      write(6,*) 'case2c'
p=(x3-x1)/a1
q=(y3-y1)/b1
if(abs(p-q).gt.tol) ipar=0
p=(x4-x1)/a1
q=(y4-y1)/b1
if(abs(p-q).gt.tol) ipar=0
endif

c ******** case 3: two of the coeffs are zero ********
c if a1=0 and b1=0 line is parallel to z axis
   if((abs(b1).lt.tol).and.(abs(a1).lt.tol)) then
      write(6,*) 'case3a'
   endif

c if a1=0 and c1=0 line is parallel to y axis
   if((abs(c1).lt.tol).and.(abs(a1).lt.tol)) then
      write(6,*) 'case3b'
   endif

c if b1=0 and c1=0 line is parallel to x axis
   if((abs(c1).lt.tol).and.(abs(b1).lt.tol)) then
      write(6,*) 'case3c'
   endif
endf

100 continue

c find intersections
   if(ipar.ne.0) then
      tee= (r1dotv1+v2dotv2-r1dotv2+v2dotv1)/v1v2sq
      if((tee.gt.0.).and.(tee.lt.1.)) then
         iflag=0
         x0=x1+tee*a1
         y0=y1+tee*b1
         z0=z1+tee*c1
   endif
c check:
    u0=x3+tee*a2
    v0=y3+tee*b2
    w0=z3+tee*c2
    if((abs(x0-u0).gt.tol).or.(abs(y0-v0).gt.tol).or.
        (abs(z0-w0).gt.tol)) iflag=1
    endif
endif
200 return
end
APPENDIX F: GEOMETRY FILE BUILDER (BLDMAT)

BLDMAT generates data files with the “.m” extension so that they can be loaded into MATLAB. The files are used by the matlab script PLTPATCH to view a three-dimensional plot of the geometry. BLDMAT also generates and edge connection list that is identical to that of PATCH.

```
c program bldmatfck.f  (DEVELOPMENT version 2: 2/96)
c program to read inpatch data (output from buildn5 or mbuild)
c and write the variables to *.m files for use by MATLAB
*******************************************************************************
  parameter(mxunkn=5500,mxbndn=2500,mxedges=5500,mxfac=3700,
    $ mxfu=5500,mxdjbd=50,mxmult=3)
*******************************************************************************
dimension x(mxbndn),y(mxbndn),z(mxbndn),node(mxbndn)
dimension nedge(mxedges),np1(mxedges),np2(mxedges),nbe(mxdjbd)
dimension datnod(3,mxbndn),nconn(3,mxedges),nbound(3,mxfac)
dimension indsum(mxedges),istart(mxdjbd+1),ipvt(mxfu)
dimension iedgf(mxmult+1,mxedges)
character=50 title
character=9 namein
multi=mxmult+1
write(6,'') 'enter input file name'
read(5,3) namein
3 format(a9)
  open(1,file=namein,status='old')
  read(1,2) title
2 format(a50)
  read(1,*) nvert,nedges
  nnodes=nvert
  write(6,'') 'nvert,nedges=',nvert,nedges
  do 10 n=1,nvert
10  read(1,*) node(n),x(n),y(n),z(n)
  do 20 n=1,nedges
20  read(1,*) nedge(n),np1(n),np2(n)
c write new data in MATLAB files
  open(12,file='xpts.m')
  open(13,file='ypts.m')
  open(14,file='zpts.m')
  open(15,file='end1.m')
  open(16,file='end2.m')
  open(3,file='facefck')
  write(3,'') namein
90 format(i15)
  do 40 n=1,nvert
    write(12,100) x(n)
    write(13,100) y(n)
  100 format(1x,1e14.6)
```

78
write(14,100) z(n)
40 continue
100 format(f15.4)
do 50 n=1,nedges
   write(15,101) np1(n)
   write(16,101) np2(n)
50 continue
101 format(i5)
c fill datnod with node locations
   do 110 n=1,nvert
       datnod(1,n)=x(n)
       datnod(2,n)=y(n)
       datnod(3,n)=z(n)
110 continue
c fill nconn with edge connections.
do 120 n=1,nedges
   nconn(1,n)=np1(n)
   nconn(2,n)=np2(n)
   nconn(3,n)=-1
120 continue
call geom(datnod,nconn,nedges,indsum,nbound,mxface,nfaces,$
   mxbdnd,nunkrb,mxedge)
call curdir(nconn,nbound,nfaces,nedges,mxjobd,ipvt,istart,$
   nbodys,nbe)
   write(6,*),'number of disjoint bodies is ',nbodys
do 210 i=1,nbodys
   call prntbd(nconn,nbound,istart,i,nedges,nfaces,nbodys)
210 continue
call edgfac(nconn,nedges,nbound,nfaces,iedgf,mult1)$
call edgep(nnodes,nedges,datnod,nconn,nunkrb$
c$get body parameters$
call bodpar(datnod,nconn,nbound,nnodes,nedges,nfaces,nbodys,$
   nunkrb)$
stop
end

===============================================
subroutine curdir(nconn,nbound,nfaces,nedges,mxjobd,itrue,istart,$
   nbodys,nbe)
===============================================
c all edges in the first disjoint surface are numbered consecutively
c starting from 1. the edges in the next disjoint surface are numbered
c consecutively, starting where the last surface left off.
c
input:
c nconn(3,nedges): edge j runs from vertex nconn(1,j) to vertex
c nconn(2,j). nconn(3,j)=multiplicity factor of the edge.
c nbound(3,nfaces): each face j has edges nbound(i,j) i=1,2,3
c j=1,2,...,nfaces.
c nffaces equals the total number of faces.
c nedges equals the total number of edges.
c mxdjbd equals the maximum number of expected bodys.
c
output:
  c istart(mxdjbd+1):istart(i)=the lowest numbered face on the ith
tree(disjoint surface)
  c istart(nbodys+1)=nffaces+1
  c mxdjbd.ge.nbodys or routine stops and prints a warning.
  c itree(nfaces)
  c itree(i) i=1,...,istart(2)-1 =the faces on the first tree.
  c itree(i) i=istart(j),...,istart(j+1)-1 =the faces on the jth tree.
  c nbodys equals the number of disjoint surfaces.
  c nbe(mxdjbd):nbe(i) contains the number of boundary edges for body i.
  c
    integer nconn(3,nedges),nbound(3,nfaces),itree(nfaces),
    > istart(mxdjbd+1),nbe(mxdjbd)
  c set present tree to first tree.
  c ntree=total number of faces stored in the tree.
  c lnf=lowest numbered face occurring in the present tree.
  c conveniently it happens that itree(lnf)=lnf so lnf=lowest index i
  c so that itree(i) is in the present tree.
    do 40 i=1,mxdjbd
      nbe(i)=0
    40 continue
    nface1=nfaces+1
    ntree=1
    lnf=1
    itree(lnf)=lnf
    istart(1)=lnf
    do 1 nbodys=1,mxdjbd
  c add the number of boundary edges in the first face of this body to
  c nbe(nbodys).
    if(nconn(3,nbound(1,lnf)).eq.0)nbe(nbodys)=nbe(nbodys)+1
    if(nconn(3,nbound(2,lnf)).eq.0)nbe(nbodys)=nbe(nbodys)+1
    if(nconn(3,nbound(3,lnf)).eq.0)nbe(nbodys)=nbe(nbodys)+1
  c search for a face that may be added to the present tree.
    do 50 iface=lnf+1,nfaces
      if iface is already in tree continue search.
        do 10 jtree=lnf,ntree
          if(iface.eq.itree(jtree))goto 50
        10 continue
    c test to see if iface has an edge in common with present tree.
      lowface=nface1
    c find the lowest face with a common edge.
      do 20 jtree=lnf,ntree
    c test for a common edge.
        do 30 i=1,3
          do 31 j=1,3
            continue
        31 continue
        do 30 continue
    50 continue
if(nboun(l,i,iface).eq.nboun(l,j,istree(jtree)).and.
   istree(jtree).lt.illowf(fe)lowface=istree(jtree)
20   continue
30   continue
31   if(lowface.ne.nfaced)then
   c common edge has been found.
       do 60 i=1,3
       do 61 j=1,3
       if(nboun(l,i,iface).eq.nbound(j,lowface))then
   c increment number of faces in the tree and add iface to present tree.
       ntree=ntree+1
       itree(ntree)=iface
   c add the number of boundary edges in this face to nbe.
       if(nconn(3,nbound(1,iface)).eq.0)
       > nbe(nbodys)=nbe(nbodys)+1
       if(nconn(3,nbound(2,iface)).eq.0)
       > nbe(nbodys)=nbe(nbodys)+1
       if(nconn(3,nbound(3,iface)).eq.0)
       > nbe(nbodys)=nbe(nbodys)+1
       goto 51
   endif
51   continue
60   continue
61   continue
50   continue
   lnf=ntree+1
   if(lnf.le.nfaced)then
       istart(nbodys+1)=lnf
   c initialize new tree.
       ntree=ntree+1
       itree(ntree)=lnf
   else
       goto 999
   endif
1   continue
   write(3,99)
99   format(1x,'warning in curdir mxdjb.found but still have faces',.
   $' left')
   stop
999   continue
   istart(nbodys+1)=nfaces+1
   return
end

======================================================================
subroutine bodpar(datnod,nconn,nbound,mnodes,nedges,nfaces,
   $ nbodys,nunknb)
======================================================================
c input:

81
c datnodi,j i=1,2,3 are the x,y,z coordinates of the jth node.
c nconn(3,j): edge j runs from node nconn(1,j) to node nconn(2,j)
c nconn(3,j) is the multiplicity of the jth edge.
c nbound(i,j): contains the ith edge of the jth face i=1,2,3.
c nnodes = the number of body nodes.
c nedges = the number of edges.
c nfacess = the number of faces.
c nunknb = the number of body unknowns.
c output:
c aveedge = the average edge length(meters**2) including multiplicity.
c edgemx = the maximum edge length(meters).
c mxedge = the edge number of the edge with length edgemx.
c edgenn = the minimum edge length(meters).
c mmedge = the edge number of the edge with length edgenn.
c tarea = the surface area of the scatter(meters**2):for thin
   structures only one side is considered in the surface area.
c avaraex = the average area of the faces.
c mmarea = the number of the face with the maximum area(areaax).
c mnarea = the number of the face with the minimum area(areaam).
c ratio = the minimum height to base ratio over all faces.
c mmratio = the face number that has a height to base ratio of 'ratio'.
  dimension datnod(3,nnodes)
c integer nconn(3,nedges),nbound(3,nfaces)
c common/params/avedge,edgemx,mxedge,edgenn,mmedge,tarea,avaarea,
c $mxarea,mnarea,areaax,areaam,ratio,mmratio
c the following line is a statement function.
  size(x,y,z)=sqrt(x*x+y*y+z*z)
c initialization.
sedgl=0
  valmax=1.e35
  valmin=-1.e35
  edgemx=valmin
  edgenn=valmax
  do 20 ie=1,nedges
    mult=nconn(3,ie)
    n1=nconn(1,ie)
    n2=nconn(2,ie)
    x=datnod(1,n2)-datnod(1,n1)
    y=datnod(2,n2)-datnod(2,n1)
    z=datnod(3,n2)-datnod(3,n1)
    edgl=size(x,y,z)
    sedgl=sedgl+mult*edgl
    if(edgl.gt.edgemx)then
      edgemx=edgl
      mxedge=ie
    endif
    if(edgl.lt.edgenn)then
      edgenn=edgl
      mmedge=ie
  20 enddo
endif
20 continue
avedge=sedgl/nunknb
c compute tarea, avarea, anarea, areamm, mxarea, areamx, ratio, and mnrtio.
ratio=valmax
areamx=valmin
areamm=valmax
tarea=0.
do 40 iface=1,nfaces
   is1=nbound(1,iface)
   is2=nbound(2,iface)
   is3=nbound(3,iface)
call fac vtx(nconn,nedges,is1,is2,is3,nv1,nv2,nv3)
call vtx crd(datnod,nnodes,nv1,nv2,nv3,x1,x2,x3,y1,y2,y3,z1,z2,z3
>)
x1mx3=x1-x3
y1my3=y1-y3
z1mz3=z1-z3
x2mx3=x2-x3
y2my3=y2-y3
z2mz3=z2-z3
x2mx1=x2-x1
y2my1=y2-y1
z2mz1=z2-z1
c compute area of face by taking the cross product of two edge vectors.
   vx=y1my3*z2mz3-z1mz3*y2my3
   vy=z1mz3*x2mx3-x1mx3*z2mz3
   vz=x1mx3*y2my3-y1my3*x2mx3
   area=.5*sqrt(vx,vy,vz)
c compute the square of the lengths of each side.
   r1s=x2mx3*x2mx3+y2my3*y2my3+z2mz3*z2mz3
   r2s=x1mx3*x1mx3+y1my3*y1my3+z1mz3*z1mz3
   r3s=x2mx1*x2mx1+y2my1*y2my1+z2mz1*z2mz1

c compute the height to base ratios.
   area2=area+area
   htb1=area2/r1s
   htb2=area2/r2s
   htb3=area2/r3s
   htbmin=amin1(htb1,htb2,htb3)
tarea=tarea+area
if(area.gt.areamm)then
   mxarea=iface
   areamx=area
endif
if(area.lt.areamm)then
   mnarea=iface
   areamm=area
endif
if(htbmin.lt.ratio)then

83
mnrtio=iface
ratio=htbmin
endif

continue

avarea=tarea/nfaces
write(3,110)

110 format(/'25x,'body parameter list'/)
write(3,111) nnodes,nedges,nfaces,nunknb,nbods

111 format(10x,'number of vertices=',i5,10x,'number of edges=',i5,10x>
>\times','number of faces=',i5,10x,'number of edges including multiplici
>\times','number of bodies=',i5)
write(3,205)

205 format(/'25x,'modeling parameter list (meters)'/)
write(3,206) tara

206 format(10x,'surface area of the scatterer=',e12.5,1x,'sq.meters')
write(3,209) aedge,mxedge,edge,nnedge,edge,edge

209 format(10x,'average edge length=',e12.5,1x,'meters',
$/10x','maximum edge length(edge no.=',i5,')=',e12.5,1x,'meters',
$/10x','minimum edge length(edge no.=',i5,')=',e12.5,1x,'meters')
write(3,210) avarea,mxarea,area,x,nnarea,area

210 format(10x,'average face area =',e12.5,1x,'sq.meters',/10x,
$/10x','maximum face area (face no.=',i5,1x,')=',e12.5,1x,'sq.meters',/,
$/10x','minimum face area (face no.=',i5,1x,')=',e12.5,1x,'sq.meters')
write(3,211) mnrtio,ratio

211 format(10x,'minimum face height to base ratio (face no.=',
$\times','i5,1x,')=',e11.5)
if((area,nn,lt,1.e-10).or.(ratio,lt,1.e-10)) then
write(6,*) 'TRIANGLES WITH ZERO AREA'
stop
endif
return
end

===============================================
subroutine facvtx(nconn,nedges,ie1,ie2,ie3,nv1,nv2,nv3)
===============================================
c nv1 is node opposite edge ie1.
c nv2 is node opposite edge ie2.
c nv3 is node opposite edge ie3.
integer nconn(3,nedges)
c the node nv1 is the node that edges 2 and 3 have in common.
c the node nv3 is the other node on edge 2.
if(nconn(1,ie2).eq.nconn(1,ie3).or.nconn(1,ie2).eq.nconn(2,ie3))
$then
  nv1=nconn(1,ie2)
nv3=nconn(2,ie2)
else
  nv1=nconn(2,ie2)
nv3=nconn(1,ie2)
endif
c the node nv2 is the node that edges 1 and 3 have in common.
    if(nconn(1,ie1).eq.nconn(1,ie3).or.nconn(1,ie1).eq.nconn(2,ie3))
$then
    nv2=nconn(1,ie1)
else
    nv2=nconn(2,ie1)
endif
return
end

subroutine vtxcrd(datnod,nnodes,n1,n2,n3,x1,x2,x3,y1,y2,y3,z1,z2,
>z3)

dimension datnod(3,nnodes)
x1=datnod(1,n1)
y1=datnod(2,n1)
z1=datnod(3,n1)
x2=datnod(1,n2)
y2=datnod(2,n2)
z2=datnod(3,n2)
x3=datnod(1,n3)
y3=datnod(2,n3)
z3=datnod(3,n3)
return
end

subroutine edgfac(nconn,nedges,nbound,nfaces,iedgf,multi)

c input:
c edge ie runs from vertex nconn(1,ie) to vertex nconn(2,ie)
c and has multiplicity nconn(3,ie).
c face iface has edges nbound(j,iface) j=1,2,3
nc multi is set in the main program and multi-1.ge.the
nc maximum multiplicity of any edge.
c output:
c array iedgf for an edge with multiplicity mult
nc iedgf(1,ie)=the lowest numbered face connected to edge ie.
c iedgf(2,ie)=the next lowest numbered face connected to ie.
c ..............
c iedgf(mm,ie)=the last face connected to ie.
c where mm is the number of faces connected to edge ie.
c integer nconn(3,nedges),nbound(3,nfaces),iedgf(multi,nedges)
c initialize the array iedgf.
   do 5 ie=1,nedges
      do 6 m=1,multi
         iedgf(m,ie)=0
    6  continue
5  continue

nc fill array iedgf.
do 100 ie=1,nedges
  multie=nconn(3,ie)+1
  m=0
  do 50 if=1,nfaces
    if(m.ge.multie)go to 100
    if(ie.eq.nbound(1,if).or.ie.eq.nbound(2,if).or.ie.eq.
$ nbound(3,if))then
      m=m+1
      iedgfm(ie,if)=if
    endif
  enddo
100 continue
write(3,*)
  do 200 ie=1,nedges
    write(3,201)ie
    write(3,*) (iedgfm(ie,if),if=1,nconn(3,ie)+1)
  enddo
200 continue
format(1x,'(edge,i5,' is attached to faces '),)
return
end

=========================================
subroutine edgep(nnod,mudege,datnod,nconn,munkn)
> ,iedgfm,nf,plbnd,unmbn)
=========================================

input:
  nnod is the number of body nodes.
  nedege is the number of edges.
  munkn is the number of body unknowns before considering the
  symmetry plane attachments.
  multi is the maximum allowed multiplicity for an edge plus 1.
  nf is the number of faces.
  datnod(i,nnod) is the x, y, z components (i=1,2,3) of the nth node n=1,nnod.
  nconn(i,ie) = i=1,2,3: edge ie (ie=1,nedege) runs from node nconn(i,ie)
  to nconn(2,ie) and has multiplicity nconn(3,ie) (before any symmetry
  plane attachments are considered).
  nboun(j,if) = j=1,2,3 are the three edges attached to face
  number if. if = 1, nf.
  iedgfm(ie,if) = i=1,...,nconn(3,ie)+1 contains the faces attached to edge
  number ie (before any symmetry plane attachments are considered).
  ie = 1,...,nedege.

output:
  for each edge ie that is connected to at least one p.e.c. symmetry plane
  the number of body unknowns (munkn) is incremented by 1.
  the edge vertex connection list with edge multiplicities is outputted
  after accounting for all symmetry plane attachments.
  dimension datnod(3,nnod),nconn(3,nedege),igndp(3),
> iedgfm(nf,nnod),plbnd,unmbn)
  common/gplane/mgndp,igndp
  logical lfpmpc
c    save /gplane/
if(ngndp.gt.0)then
  c if the maximum distance from an edge to a symmetry plane is less than or
c equal to edged, the edge is assumed connected to that symmetry plane.
edged=1e-7
  do 100 ie=1,nedges
    n1=nconn(1,ie)
    n2=nconn(2,ie)
    x1=abs(datnod(1,n1))
    y1=abs(datnod(2,n1))
    z1=abs(datnod(3,n1))
    x2=abs(datnod(1,n2))
    y2=abs(datnod(2,n2))
    z2=abs(datnod(3,n2))
    xm=amax1(x1,x2)
    ym=amax1(y1,y2)
    zm=amax1(z1,z2)
    ix=0
    iy=0
    iz=0
    if(xm.le.edged)ix=igndp(1)
    if(ym.le.edged)iy=igndp(2)
    if(zm.le.edged)iz=igndp(3)
    npes=-amin0(iy,0)+amin0(iz,0)+amin0(iy,0)
    npes=amin0(iy,0)+amin0(iz,0)
    if(npes.ge.1)then
      c case edge is attached to at least pec
      nunknb=nunknb+1
      nconn(3,ie)=nconn(3,ie)+1
    endif
  100   continue
endif
write(3,29)
29  format(14x,'edge-vertex connection list/)
do 40 i=1,nedges
  write(3,331)i,nconn(1,i),nconn(2,i),nconn(3,i)
40   continue
331  format(3x,'edge',i5,' goes from vertex','i5,' to vertex','i5,
$' mult='i3)
return
end

subroutine prnthnd(nconn,nbound,istart,i,nedges,nfaces,nbodys)
This subroutine prints the edges and the vertices of each face.
input:
c nconn has the vertices and the multiplicity factor for each edge.
c nbound has the edges for each face.
c istart has the beginning faces for each body.

87
c i is the present body.
c nedges is the total number of edges.
c nfaces is the total number of faces.
c nbodys is the total number of bodys.
    integer nconn(3,nedges),nbound(3,nfaces),istart(nbodys+1)
    integer nverts(10000,3)
    open(20,file='node1.m')
    open(21,file='node2.m')
    open(22,file='node3.m')
c loop through the faces of this body.
do 10 i10=istart(i),istart(i+1)-1
    call facvtx(nconn,nedges,nbound(1,i10),nbound(2,i10),
        >nbound(3,i10),nv1,nv2,nv3)
    write(3,98)i10,nbound(1,i10),nbound(2,i10),nbound(3,i10),nv1,nv2
        >,nv3
    c write face number and vertices to 'facedat'
    c write(9,*)i10,nv1,nv2,nv3
    write(20,100) nv1
    write(21,100) nv2
    write(22,100) nv3
    nverts(i10,1)=nv1
    nverts(i10,2)=nv2
    nverts(i10,3)=nv3
10 continue
98 format(1x,'face',i5,' has edges',i5,' with vertices',i5)
100 format(i5)
    return
end

Subroutines GEOM, FACETCK, and AXB are used here. Listings appear in Appendix A.
APPENDIX G: MATLAB GEOMETRY VIEWER (PLTPATCH)

PLTPATCH uses the data files produced by BLDMAT to plot the "inpatch" geometry. PLTPATCH has flags that can be changed by the user to choose between wire grid, 3D, and 3D with hidden lines.

% Program to read "buildn5" data and plot it in MATLAB
% node (x,y,z) coordinates
clear
cif
% legend:
% icur=0: plot wire grid
% icur=1: surface plot (with hidden lines)
% icur=2: surface plot with current intensities
% icur=3: plot currents for a "deck walk"
icur=0;
load xpts.m
load ypts.m
load zpts.m
nverts=length(xpts);
% node connection list
load end1.m
load end2.m
nedges=length(end1);
% face connection list
load node1.m
load node2.m
load node3.m
nfaces=length(node3);
% load vind array which gives the three nodes for each triangle
for i=1:nfaces
    pts=[node1(i) node2(i) node3(i)];%vind(i,:)=pts;
end
% Read currents if intensities are to be plotted
if icur==2
    load currents.m
    currents=10*log10(currents+1.e-10);
    minc=min(currents);
    maxc=max(currents);
    currents=(currents-minc)/(maxc-minc);
end
% This section plots a mesh
if icur==0
    for i=1:nfaces
        X=[xpts(vind(i,1)) xpts(vind(i,2)) xpts(vind(i,3)) xpts(vind(i,1))];
        Y=[ypts(vind(i,1)) ypts(vind(i,2)) ypts(vind(i,3)) ypts(vind(i,1))];
        Z=[zpts(vind(i,1)) zpts(vind(i,2)) zpts(vind(i,3)) zpts(vind(i,1))];
plot3(X,Y,Z,'m')
if i == 1
    axis equal
view(60,45)
    hold on
end
end
end

% This section plots surfaces (therefore can use hidden)
if icur==0
    colormap(hot)
colormap(currents*64);
it=0;
for i=1:nfaces
    it=it+1;
    X=xpts(vind(i,1)) xpts(vind(i,1)); xpts(vind(i,2)) xpts(vind(i,3))];
    Y=ypts(vind(i,1)) ypts(vind(i,1)); ypts(vind(i,2)) ypts(vind(i,3))];
    Z=zpts(vind(i,1)) zpts(vind(i,1)); zpts(vind(i,2)) zpts(vind(i,3))];
end
% Save handles for each triangle
if it==1
    axis equal
view(60,45)
    hold on
end
if icur==2
    C=[floor(currents(i)) floor(currents(i)) ; ...
    floor(currents(i)) floor(currents(i))];
    spl[it]=surf(X,Y,Z,C);
end
if icur==1
    spl[it]=surf(X,Y,Z);
    set(spl[it],'facecolor','black');
end
% fill3(X,Y,Z,C);
% patch(x,y,z)
    set(spl[it],'edgecolor','white');
end
end
% label nodes if desired
ilabn=1;
if ilabn==0
    for i=1:nverts
        text(xpts(i),ypts(i),zpts(i),num2str(i))
    end
end
% label edges
ilabe=0;
if ilabe==0

90
for i=1:nedges
    xav=(xpts(end1(i))+xpts(end2(i)))/2.;
    yav=(ypts(end1(i))+ypts(end2(i)))/2.;
    zav=(zpts(end1(i))+zpts(end2(i)))/2.;
    text(xav,yav,zav,num2str(i))
end
end
axis square
xlabel('x')
ylabel('y')
ylabel('z')
hold off
delx=max(xpts)-min(xpts);
dely=max(ypts)-min(ypts);
delz=max(zpts)-min(zpts);
del=max([delx dely delz]);
axis([min(xpts),min(xpts)+del,min(ypts),min(ypts)+del,...
     min(zpts),min(zpts)+del])
## INITIAL DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>No.</th>
<th>Distribution</th>
<th>Copies</th>
</tr>
</thead>
</table>
| 1   | Defense Technical Information Center  
     8725 John J. Kingman Road, STE 0944  
     Ft. Belvoir, VA 22060-6218         | 2      |
| 2   | Dudley Knox Library  
     Naval Postgraduate School  
     411 Dyer Road  
     Monterey, CA 93943-5101            | 2      |
| 3   | Chairman, Code EC  
     Department of Electrical and Computer Engineering  
     Naval Postgraduate School  
     833 Dyer Road, Room 437  
     Monterey, CA 93943-5121            | 1      |
| 4   | Prof. David C. Jenn, Code EC/Jn  
     Department of Electrical and Computer Engineering  
     Naval Postgraduate School  
     833 Dyer Road, Room 437  
     Monterey, CA 93943-5121            | 3      |
| 5   | Prof. Jeffrey B. Knorr, Code EC/Ko  
     Department of Electrical and Computer Engineering  
     Naval Postgraduate School  
     833 Dyer Road, Room 437  
     Monterey, CA 93943-5121            | 1      |
| 6   | Prof. Richard W. Adler, Code EC/Ab  
     Department of Electrical and Computer Engineering  
     Naval Postgraduate School  
     833 Dyer Road, Room 437  
     Monterey, CA 93943-5121            | 1      |
| 7   | Prof. Charles Calvano, Code ME/Ca  
     Department of Mechanical Engineering  
     Naval Postgraduate School  
     Monterey, CA 93943-5146             | 1      |
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
</tr>
</thead>
</table>
| 8   | CAPT Charles Ristorcelli | Commander  
SPAWAR PMW 163  
2451 Crystal Park 5  
Arlington, VA 22245-5200 |                |
| 9   | CAPT Roger Connell     | SPAWAR PMW 163  
2451 Crystal Park 5  
Arlington, VA 22245-5200 | 1              |
| 10  | Ralph Skiano           | SPAWAR PMW 163  
2451 Crystal Park 5  
Arlington, VA 22245-5200 | 1              |
| 11  | Gary Wang              | SPAWAR PMW 163  
2451 Crystal Park 5  
Arlington, VA 22245-5200 | 1              |
| 12  | Jeffrey Lucas, Code 372JL | NISE-East  
400 Marriot Drive  
North Charleston, SC 29406-6504 | 1              |
| 13  | LCDR Roger McGinnis    | NAVSEA, Code SEA-03T1  
2531 Jefferson Davis Highway  
Arlington, VA 22242 | 1              |
| 14  | Dr. S.T. Li            | NRAD, Code 824  
53225 Millimeter Street  
San Diego, CA 92152-5000 | 1              |
| 15  | Jay Rockway            | NRAD, Code 824  
53225 Millimeter Street  
San Diego, CA 92152-5000 | 1              |
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Company</th>
<th>Address</th>
<th>City, State, Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Randy Ott</td>
<td>SWL Inc.</td>
<td>3900 Juan Tabo, NE</td>
<td>Albuquerque, NM 87111</td>
</tr>
<tr>
<td>17</td>
<td>Dave Hovey</td>
<td>Mantech Systems</td>
<td>16541 Commerce Drive, Suite 1</td>
<td>King George, VA 22485</td>
</tr>
<tr>
<td>18</td>
<td>Norm Saucier</td>
<td>Lockheed Sanders</td>
<td>PTP1-1813</td>
<td>Nashua, NH 03061-0868</td>
</tr>
</tbody>
</table>