Mercury Method of Moments Adjunct Visualization Tool: Trials and Tribulations

Daniel Topa
ERT Corp
Kirtland AFB
Albuquerque, NM

April 14, 2020

Abstract

While `Mercury MoM` has successfully modeled radar cross sections to test objects, the 64-bit adjunct tool, `MM Viz` has been problematic in the Windows 10 environment and most popular Linux distributions. One purpose of this report is to describe the purpose of the visualization tool as a pre– and post–processor for `Mercury MoM` to document the efforts to use it. Another purpose is to show that the intrinsic tool of mesh repair has a modern replacement and that the MATLAB scripts are replaced by Python scripts.

1 `Mercury MoM`

The Fortran code developed by John Shaeffer and called Mercury Method of Moments is an elegant tool for modeling electromagnetic interactions based upon SIE and VIE. However, years of feature creep have yielded a code which is cumbersome and brittle. The core mathematics remain crisp; the code has an impressive ability to solve systems of equations with millions of unknowns. In a personal phone call with the author, he described his desire to rewrite MMoM as an open source code, exploiting significant improvements in the Fortran language, and to target HPC users, breaking free of the constraints of the desktop box. But that is a tale for another day.

In the beginning, NASA, the corporate sponsor for `Mercury MoM`, supported a separate effort to expand the user base for MMoM by providing a free tool for model creation. This was in a time before the cloud, before the rise of open source software. An adjunct visualization package, `MM Viz`, was created many years ago to provide a basic mesh generation tool to create models suitable for `Mercury MoM` (MMoM). The development and maintenance of `MM Viz` fell to Kom Ham.

John expressed some frustration with his NASA sponsor, feeling a disconnect between his desire to write a robust code and the funders who wanted new features. In retirement, he is crafting a new version based upon current Fortran and is thinking about open source release.

While the computation engine of `Mercury MoM` has been tested and validated with some

vigor,[1] the visualization tool has escaped rigorous examination.

2 MM Viz

2.1 An Outdated Adjunct

MM Viz is the complicated complement to Mercury MoM. Invaluable when first released, the tool has fallen behind other open source projects. While it is the only tool directly mated to Mercury MoM the specific tasks it performed have been replicated with open source tools like Python. In point of fact, MM Viz also relied upon the propriety MATLAB package, and the AFIT package ALPINE. Both MATLAB and ALPINE have been obviated by improvements to the open source FreeCAD application.

Table 1 lists the critical functions needed by Mercury MoM and catalogs how these tasks were completed by Capt. Sciacca and by Chris McGeorge.

Table 1: Auxiliary functions performed by packages external to Mercury MoM and the McGeorge mitigations.

Function	Sciacca	McGeorge
*.obj ⇒ *.facet	MATLAB/ALPINE	Python
Sealing FreeCAD mesh	MM Viz	FreedCAD
Plotting RCS values	MATLAB/ALPINE	Python

Python scripts were promptly written to bypass the need for ALPINE and its dependence upon MATLAB. Moving from a proprietary code base to an open source one is a prudent choice in software management given the trivial time expense for crafting the replacement Python routines.

So the primary need for MM Viz was sealing the mesh. This means insuring that the mesh completing covers the CAD model with neither under– nor overlap. McGeorge pointed out that FreeCAD will perform this task admirably, as seen by the sophisticated suite of repair tools in Figure 1. There was a period of angst where we thought this mesh repair did not work. But we have now demonstrated that the problem is intrinsic to the single precision arithmetic in Mercury MoM.

2.2 Frustrations with MM Viz

MM Viz evidenced a broad spectrum of failure in the Windows 10 and most popular Linux environments. The Windows versions either froze or crashed outright, the Linux versions typically through a segmentation fault upon launch. The depth and breadth of failure was sobering.

We had access to three different versions of the tool:

1. 4.0.9,
2. 4.1.0,
3. 4.1.12.

Caption Joe Sciacca, who successfully used MM Viz in a 32-bit Windows 7 environment, almost certainly used the two older versions.

Our efforts to use MM Viz centered around two different tutorials.

[1] *Mercury Method of Moments: Code Validation Test Cases*, Kam Hom, NASA/LDTM-201502; *Mercury Method of Moments: Benchmark Study of Performance, Memory Requirements, Accuracy*, Kam Hom, NASA/LDTM-201504

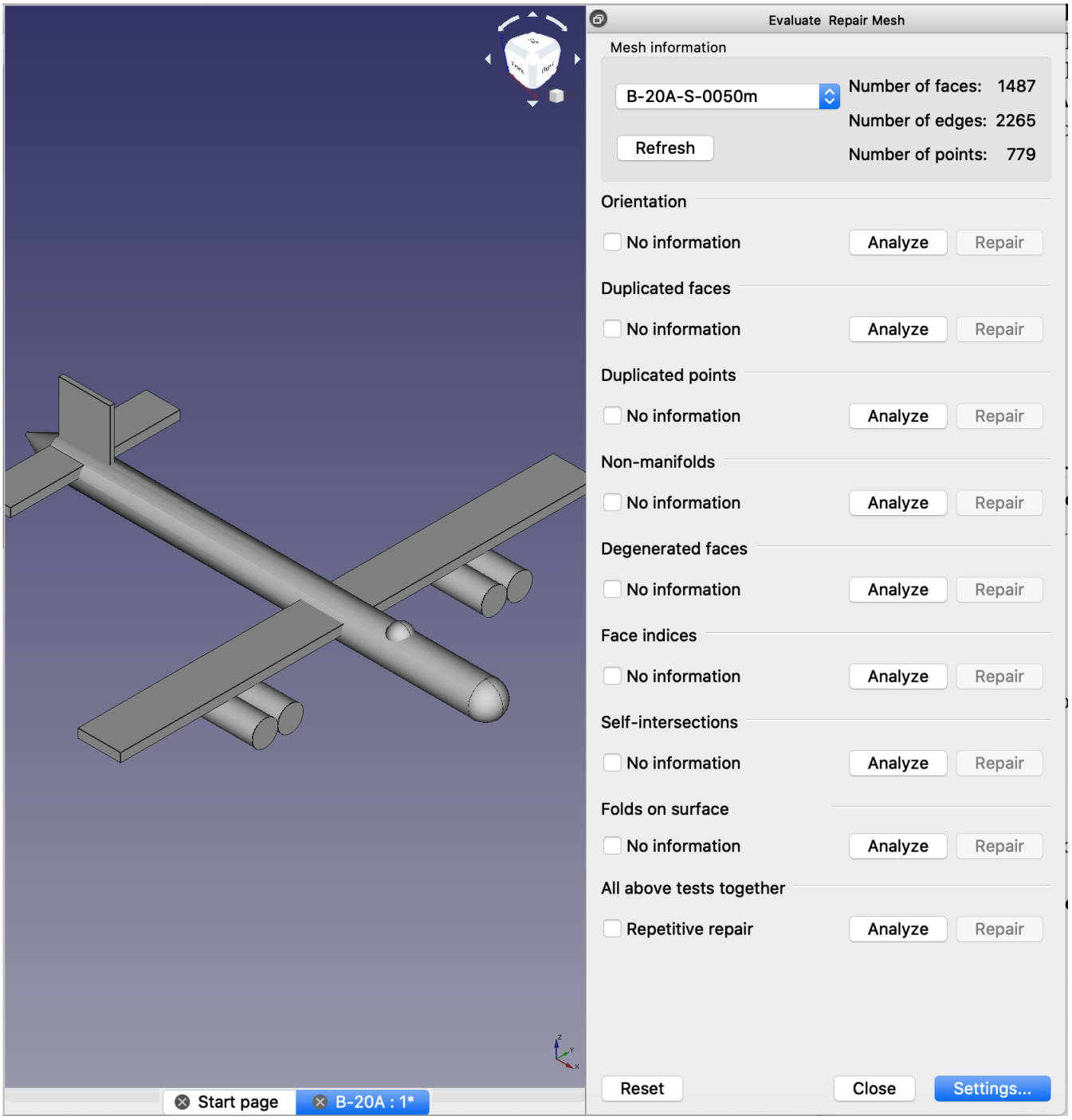

Figure 1: The suite of mesh repair tools in FreeCAD preferred by Chris McGeorge.

1. Pill tutorial
2. AFIT Sphere tutorial

The first is from a tutorial manual[2] provided with the distribution. The second is based upon the AFIT demonstration video `sphereTutorial` provided by Capt. Sciacca.

We explored several different desktop and laptop machines, with different environments:

1. Windows 10
   (a) Native (Dan, Chris, Trevor, Eric)
   (b) Virtual Box (Build 18362.h)
2. Linux OS
   (a) Centos 7
   (b) Fedora 3.1
   (c) Ubuntu 20, 19
   (d) Scientific Linux 6
   (e) BSD

All to the same end: crash, freeze, set fault, etc.

Of note is a clever automation tool developed by Chris McGeorge which runs through the GUI version of `MM Viz`. This insightful tool guarantees a reproducible task sequence and allows one to experiment with the Windows 10 environment (e.g. rebooting).

2.3 `MM Viz` Diagnostics

Along the way, we cataloged a few diagnostics on `MM Viz`, using the `gdb` backtrace, the `ldd` and `objdump` tools.

2.3.1 Backtrace

Despite not being compiled with debug symbols (via the `-g` flag), there is still useful information to be harvested from at backtrace of the core dump using the GNU debugger `gdb`. The following backtrace listing tells (`gdb -ex bt ./MMViz_4.1.12 core`) us that

1. Line 25: `Program terminated with signal SIGSEGV, Segmentation fault.`
2. Line 26: error occurred in the subroutine `vector_and_utility_module_mp_real_vector_norm_.A`
3. Subsequent lines show the call stack

[2] *Mercury Method of Moments: Pill Tutorial*, Kam Hom, NASA/LDTM—201501

```
dantopa@dtopa-latitude-5491:bin $ gdb -ex bt ./MMViz_4.1.12 core

GNU gdb (Ubuntu 9.0.90.20200105-0ubuntu1) 9.0.90.20200105-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./MMViz_4.1.12...
(No debugging symbols found in ./MMViz_4.1.12)
[New LWP 1885649]
[New LWP 1885710]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `./MMViz_4.1.12'.
Program terminated with signal SIGSEGV, Segmentation fault.
--Type <RET> for more, q to quit, c to continue without paging--c
#0  0x00000000042730b in vector_and_utility_module_mp_real_vector_norm_.A ()
[Current thread is 1 (Thread 0x7f4673884e00 (LWP 1885649))]
#0  0x00000000042730b in vector_and_utility_module_mp_real_vector_norm_.A ()
#1  0x0000000000545b78 in sie_geometry_module_mp_sie_geometry_tri_compute_.A ()
#2  0x0000000000643b1d in mmviz_geometry_module_mp_readgeometry_.A ()
#3  0x0000000000746e37 in MMViz::loadFile(QString const&) ()
#4  0x0000000000757004 in MMViz::qt_metacall(QMetaObject::Call, int, void**) ()
#5  0x00007f46754c8f3b in QMetaObject::activate(QObject*, int, int, void**) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtCore.so.4
#6  0x00000000007567c4 in currentUI::loadFile(QString) ()
#7  0x00000000006a9dbd in currentUI::createGeometry() ()
#8  0x0000000000756428 in currentUI::qt_metacall(QMetaObject::Call, int, void**) ()
#9  0x00007f46754c8f3b in QMetaObject::activate(QObject*, int, int, void**) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtCore.so.4
#10 0x00007f46760b3fc9 in QAbstractButtonPrivate::click() () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#11 0x00007f46760b418b in QAbstractButton::mouseReleaseEvent(QMouseEvent*) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#12 0x00007f4675e9fc2f in QWidget::event(QEvent*) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#13 0x00007f4675e6d599 in QApplicationPrivate::notify_helper(QObject*, QEvent*) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#14 0x00007f4675e6cef8 in QApplication::notify(QObject*, QEvent*) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#15 0x00007f4675eb4095 in QETWidget::translateMouseEvent(_XEvent const*) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#16 0x00007f4675ead60f in QApplication::x11ProcessEvent(_XEvent*) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#17 0x00007f4675ec5e45 in QEventDispatcherX11::processEvents(QFlags<QEventLoop::ProcessEventsFlag>)
…() from /home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtGui.so.4
#18 0x00007f46754b5be7 in QEventLoop::processEvents(QFlags<QEventLoop::ProcessEventsFlag>) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtCore.so.4
#19 0x00007f46754b5d17 in QEventLoop::exec(QFlags<QEventLoop::ProcessEventsFlag>) () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtCore.so.4
#20 0x00007f46754b92cd in QCoreApplication::exec() () from
…/home/dantopa/RCS-project/4.1.12/Linux64/bin/libQtCore.so.4
#21 0x0000000000720479 in main ()
```

2.3.2 Complexity

There is a stark contrast between the library dependencies of `MM Viz` and `Mercury MoM`. `MM Viz` relies upon several more libraries, and these libraries are considerably larger. These libraries can be cataloged with either `ldd` or as here, with `objdump`. Tools shipped exclusively for the `MM Viz` executable table are:

1. Intel Fortran
2. Intel Math Kernel Library (MKL)
3. Intel Math Library (`libimf.so`)
4. Intel optimized version of the system rand48() generator family (`libirng.so`)
5. Qt (cross-platform C++ framework)
6. Support vector classification (`libsvm`)

`MM Viz` also relies upon a richer spectrum of system libraries including

1. FreeType fontrendering
2. OpenGL high performance graphics
3. X Window System

Table 2 compare the library dependencies of both `MM Viz` and `Mercury MoM`, using for example `$ objdump -p mmv_4.1.12 | grep NEEDED`. The table makes the immediate point that the visualization tool is the far more complex one.

2.4 Possible error source

The sources of bugs are legion here, and we select just a typical Intel bug report:

> **Problem Description**: An unexpected segmentation fault may be seen at runtime start-up when an application built with these *Intel compiler* versions is run on a system containing a *recent* `libc.so.6`.[3]
>
> **Cause**: This appears to be due to a symbol conflict between `libintlc.so.5` from the Intel compiler version 16.0.3 or 16.0.4 and libc.so.6.
>
> **Workaround**: It may be worked around by replacing `libintlc` by the corresponding library from the version 17 compiler, or possibly by preloading `libintlc.so.5` from the version 17 compiler.

3 Conclusion

The success of Capt. Sciacca using `MM Viz` in the Windows 7 environment is undeniable. So too is the need to use the more modern mesh tools in FreeCAD and to move beyond the MATLAB/ALPINE toolset into the open source world.

4 Attachements

The three most critical routines we need from ALPINE have been extracted and attached as PDF documents.

1. `obj2facet.m`: converts the FreeCAD `*.obj` file into the `Mercury MoM`-friendly `*.facet` format.
2. `plotRCS.m`: harvests the electric field data from the `*.4112.txt` file and plots radar cross section.
3. `plotRange.m`: another RCS plotting tool.

[3] The library `libintlc.so.5` is noted in table 2, line 15.

Table 2: Compare the complexity of the two applications: the libraries included with the distribution (above the blue line) are exclusively for `MM Viz`. `Mercury MoM` uses standard system libraries. Libraries above the line are provided with the distribution.

		MM Viz	Mercury MoM
1	NEEDED	libifcore.so.5	
2	NEEDED	libifport.so.5	
3	NEEDED	libimf.so	
4	NEEDED	libintlc.so.5	
5	NEEDED	libiomp5.so	
6	NEEDED	libirng.so	
7	NEEDED	libmkl_core.so	
8	NEEDED	libmkl_intel_lp64.so	
9	NEEDED	libmkl_intel_thread.so	
10	NEEDED	libQtCore.so.4	
11	NEEDED	libQtGui.so.4	
12	NEEDED	libQtOpenGL.so.4	
13	NEEDED	libsvml.so	
14	NEEDED	libstdc++.so.6	
15	NEEDED	libc.so.6	libc.so.6
16	NEEDED	libdl.so.2	libdl.so.2
17	NEEDED	libfontconfig.so.1	
18	NEEDED	libfreetype.so.6	
19	NEEDED	libgcc_s.so.1	libgcc_s.so.1
20	NEEDED	libGL.so.1	
21	NEEDED	libGLU.so.1	
22	NEEDED	libICE.so.6	
23	NEEDED	libm.so.6	libm.so.6
24	NEEDED	libpthread.so.0	libpthread.so.0
25	NEEDED	libSM.so.6	
26	NEEDED	libX11.so.6	
27	NEEDED	libXext.so.6	
28	NEEDED	libXinerama.so.1	
29	NEEDED	libXrandr.so.2	
30	NEEDED	libXrender.so.1	
31	NEEDED		ld-linux-x86-64.so.2

```
function [ ] = obj2facet( varargin )
%OBJ2FACET Converts 'Alias Mesh (*.obj)' to 'ACAD facet (*.facet)' file
%   The conversion function reads the mesh file exported from FreeCAD and
%   generates a triangular facet file in the ACAD facet mesh format.
%
%INPUT: aliasMeshFilename - filename of *.obj file (include path if needed)
%       facetMeshFilename - filename of *.facet file (saved in same place)
%       (all inputs are optional - same filename used if not specified)

% Parse inputs ------------------------------------------------------------
if nargin<1||~exist(varargin{1},'file')
    [name,pathstr] = uigetfile('*.obj');
    if name==0;
        disp('obj2facet ERROR:  Input file read error - aborting.  ');
        return
    else
        ext = name(end-3:end);
        name = name(1:end-4);
    end
else
    [pathstr,name,ext] = fileparts(varargin{1});
    varargin(1) = [];
end

% Open and read *.obj file ------------------------------------------------
fid = fopen(fullfile(pathstr,[name ext]));
tline = fgetl(fid);  % read first line...
if isempty(strfind(tline,'#'))  % if it is a comment line...
    fseek(fid, 0, 'bof');  % skip it.
end
data = textscan(fid,'%s%n%n%n');
fclose(fid);

% Create vertex and facet arrays ------------------------------------------
v = [data{2}(strcmp('v',data{1})) data{3}(strcmp('v',data{1}))...
     data{4}(strcmp('v',data{1}))];
f = [data{2}(strcmp('f',data{1})) data{3}(strcmp('f',data{1}))...
     data{4}(strcmp('f',data{1}))];
vNum = size(v,1);
fNum = size(f,1);

% Open and write the *.facet file in ACAD facet file format ---------------
if nargin<1||isempty(varargin{1},'file')
    fid = fopen(fullfile(pathstr,[name '.facet']),'w');
else
    fid = fopen(varargin(1));
end

fprintf(fid,'%s \n',['FACET FILE V3.4   ' datestr(now)]);  % Rev Data Time
fprintf(fid,'%d \n',1);  % Number of parts (forced to one part)
fprintf(fid,'%s \n',['<' name ' MeshModel>']);  % Part name
fprintf(fid,'%d \n',0);  % Part not mirrored
fprintf(fid,'%d \n',vNum);  % Number of vertices
for row = 1:vNum
    fprintf(fid,'%12.6f %12.6f %12.6f \n',v(row,:));  % X Y Z of vertex
end
fprintf(fid,'%d \n',1);  % Number of sub-parts (forced to one part)
fprintf(fid,'%s \n',['<' name ' MeshSheet>']);  % Sub-Part name
fprintf(fid,'%d %d %d %d %d %d %d \n',...
             [3 fNum 0 0 0 0 0]);  % Element description
for row = 1:fNum
    fprintf(fid,'%d %d %d %d \n',[f(row,:) 1]);  % Vertices and ICoat
end

fclose(fid);

end
```

```
function plotRCS(varargin)
% <<< Part of the ALPINE Code Suite >>>
% Revision: See "version.log" for revisions
% Version:  1.0  Original version, Dec 2007
% Copyright 2007, Peter J. Collins
% Air Force Institute of Technology
%
% plotRCS(varargin)
%
% FUNCTION:
% Creates standard AFIT pattern cut plot
% Can also plot RCS as function of position
%
% INPUTS:
% An arbitrary number of AFIT rcs structures
% NOTE:  To plot polar format, the last structure should be followed by the string 'polar'
%        To set magnitude(phase) range follow with string and vector 'raxis',[rhoMin rhoMax] in dBsm
%        To plot the RCS phase, the last structure should be followed by the string 'phase'
%        To unwrap the RCS phase, the last structure should be followed by the string 'unwrap'
%        To only plot co-pol data, the last structure should be followed by the string 'copol'
%        To only plot cr-pol data, the last structure should be followed by the string 'crpol'
%        To plot particular pol, the last structure should be followed by the string 'tt', 'pp',
… 'tp', or 'pt'
%        To plot timed data, the last structure should be followed by the string 'time'
%
% OUTPUTS:
% none

% The AFIT data structure contains the following matrices...
%   out.frq (frequency vector in GHz)
%   out.ph (phi observation angle vector in degrees)
%   out.th (theta observation angle vector in degrees)
%   out.x (string system x axis position in meters)
%   out.y (string system y axis position in meters)
%   out.z (string system z axis position in meters)
%   out.roll (string system roll in degrees)
%   out.pitch (string system pitch in degrees)
%   out.yaw (string system yaw in degrees)
%   out.rng (down range distance vector in meters or time vector in seconds)
%   out.xrng (cross range distance vector in meters)
%   out.time (time vector in format YYMMDDHHMMSS)
%   out.tt (complex IQ phasor array for theta/theta polarization or TD data)
%   out.pp (complex IQ phasor array for phi/phi polarization)
%   out.tp (complex IQ phasor array for theta/phi polarization)
%   out.pt (complex IQ phasor array for phi/theta polarization)
%   out.header (structure containing file information)
% Note:  The out in dBsm is obtained by 20log10(abs(iq))
%        The phase in degrees is obtained by rad2deg(angle(iq))
%        Array format [freq <az> x angle <el>]

%% Add current directory to matlab's search path
currentDir = cd;
addpath(currentDir);
% Put any support *.m files in the resource directory
if exist(strcat(currentDir,'\resource'),'dir');
    addpath(strcat(currentDir,'\resource'));
    addpath(strcat(currentDir,'\resource\hipRCS'));
    addpath(strcat(currentDir,'\resource\facetPlot'));
end

%% Check for errors
if ~isstruct(varargin{1}) % Check for rcs structure
    disp('plotRCS ERROR:  Input does not contain an RCS structure - aborting.  ')
    return
end
```

```
65
66  %% Initialize variables
67  n = 0;
68  X = {[]};
69  Y = {[]};
70  Label = {[]};
71  plotText = [];
72  angleConv = 1;
73  plotType = 'magnitude';
74  xlabelText = [];
75  ylabelText = 'RCS (dBsm)';
76  copol = 0;
77  crpol = 0;
78  polStr = [];
79  wrap = 1;
80  timeFlag = 0;
81  % Check for polar, phase, nogrid, copol and/or unwrap plot options
82  while ~isstruct(varargin{end})
83      if strcmp(varargin{end},'polar')  % Check if polar format desired
84          angleConv = pi/180;
85          varargin(end) = [];
86      elseif strcmp(varargin{end-1},'raxis')||strcmp(varargin{end-1},'caxis')  % Check for plot
  … magnitude(phase) range (dBsm)
87          rhoMin = varargin{end}(1);
88          rhoMax = varargin{end}(2);
89          varargin(end) = [];  varargin(end) =  [];
90      elseif strcmp(varargin{end},'phase')  % Check if phase plot desired
91          plotType = 'phase';
92          ylabelText = 'RCS phase (\circ)';
93          varargin(end) = [];
94      elseif strcmp(varargin{end},'copol')
95          copol = 1;
96          varargin(end) = [];
97      elseif strcmp(varargin{end},'crpol')
98          crpol = 1;
99          varargin(end) = [];
100     elseif strcmp(varargin{end},'tt')  % Check if tt-pol only option desired
101         polStr = 'tt';
102         varargin(end) = [];
103     elseif strcmp(varargin{end},'pp')  % Check if pp-pol only option desired
104         polStr = 'pp';
105         varargin(end) = [];
106     elseif strcmp(varargin{end},'tp')  % Check if tp-pol only option desired
107         polStr = 'tp';
108         varargin(end) = [];
109     elseif strcmp(varargin{end},'pt')  % Check if pt-pol only option desired
110         polStr = 'pt';
111         varargin(end) = [];
112     elseif strcmp(varargin{end},'unwrap')
113         wrap = 0;
114         varargin(end) = [];
115     elseif strcmp(varargin{end},'time')
116         timeFlag = 1;
117         varargin(end) = [];
118     else
119         disp('plotRCS ERROR:  Invalid plot parameter - aborting.  ')
120         return
121     end
122 end
123 stringFlag = strcmp(varargin{1}.header.DATATYPE,'String data');
124 if ~max(strcmp('DRIVE',fieldnames(varargin{1}.header)))  % Handles legacy data
125     varargin{1}.header.DRIVE = 'Pylon';
126 end
127
128 %% Loop through input structure building plot arrays
```

```
129 for k = 1:length(varargin)
130     % Check for errors
131     if ~isstruct(varargin{k})  % Check if contains AFIT rcs structure
132         disp('plotRCS ERROR:  Input argument not AFIT rcs structure - aborting.  ')
133         return
134     elseif ~isempty(varargin{k}.rng)  % Check if contains image data
135         disp('plotRCS ERROR:  Input argument contains image data, try plotRange - aborting.  ')
136         return
137     elseif length(varargin{k}.ph)==1&&length(varargin{k}.th)==1  % Check if structure contains angle
…  data and is single frequency
138         disp('plotRCS ERROR:  RCS file not a pattern cut, try plotFrequencySweep() - aborting.  ')
139         return
140     elseif length(varargin{k}.ph)>1&&length(varargin{k}.th)>1  % Check if spherical plot
141         disp('plotRCS ERROR:  RCS file a function of both angles, try plotSphericalRCS() - aborting.
…  ')
142         return
143     elseif length(varargin{k}.frq)>1  % Check for single frequency
144         disp('plotRCS ERROR:  More that one frequency, try plotGlobalRCS() - aborting.  ')
145         return
146     end
147     % Build X, Y, and Label matrices looping through polarizations
148     for pol = {'tt' 'pp' 'tp' 'pt'}
149         % Ignore if not co-pol, cross-pol or particular pol
150         if copol&&(strcmp(pol,'tp')||strcmp(pol,'pt')); continue; end
151         if crpol&&(strcmp(pol,'tt')||strcmp(pol,'pp')); continue; end
152         if ~isempty(polStr)&&~strcmp(pol,polStr); continue; end
153         % Ignore if no data
154         if isempty(varargin{k}.(char(pol))); continue; end
155         % Add polarization
156         n = n+1;
157         if stringFlag % Function of position (string data)
158             if strcmp(varargin{1}.header.DRIVE,'X axis')
159                 X(n) = {varargin{k}.x(:)};
160                 xlabelText = ' position (inches)';
161             elseif strcmp(varargin{1}.header.DRIVE,'Y axis')
162                 X(n) = {varargin{k}.y(:)};
163                 xlabelText = ' position (inches)';
164             elseif strcmp(varargin{1}.header.DRIVE,'Z axis')
165                 X(n) = {varargin{k}.z(:)};
166                 xlabelText = ' position (inches)';
167             elseif strcmp(varargin{1}.header.DRIVE,'Roll')
168                 X(n) = {varargin{k}.roll(:)};
169                 xlabelText = ' angle (\circ)';
170             elseif strcmp(varargin{1}.header.DRIVE,'Pitch')
171                 X(n) = {varargin{k}.pitch(:)};
172                 xlabelText = ' angle (\circ)';
173             elseif strcmp(varargin{1}.header.DRIVE,'Yaw')
174                 X(n) = {varargin{k}.yaw(:)};
175                 xlabelText = ' angle (\circ)';
176             end
177             ang = ')';
178         elseif timeFlag  % Function of measurement time
179             X(n) = {(varargin{k}.time(:)-varargin{k}.time(1))*24*60};
180             xlabelText = ' sample time (minutes)';
181             ang = ')';
182         elseif length(varargin{k}.ph)>1  % Function of azimuth
183             X(n) = {varargin{k}.ph(:)*angleConv};
184             ang = [', \theta = ' num2str(max(varargin{k}.th(:))) '\circ)'];
185             xlabelText = ' angle (\circ)';
186         else  % Function of elevation
187             X(n) = {varargin{k}.th(:)*angleConv};
188             ang = [', \phi = ' num2str(max(varargin{k}.ph(:))) '\circ)'];
189             xlabelText = ' angle (\circ)';
190         end
191         if strcmp(plotType,'phase')
```

```
            if ~wrap
                Y(n) = {180*unwrap(angle(varargin{k}.(char(pol))(:)))/pi};
            else
                Y(n) = {180*(angle(varargin{k}.(char(pol))(:)))/pi};
            end
        else
            Y(n) = {20*log10(abs(varargin{k}.(char(pol))(:)))};
        end
        Label(n) = strcat(varargin{k}.header.FILENAME,' (',pol,'-pol)');
        plotText = strcat(plotText,'X{',num2str(n),'},Y{',num2str(n),'},');
    end
end
Title = ['Pattern Cut (Frequency = ',num2str(varargin{1}.frq),' GHz',ang];
xlabelText = [varargin{1}.header.DRIVE xlabelText];

%% Create plot (based on figure generated by MATLAB)
%CREATEFIGURE1(X1,YMATRIX1)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data

%  Auto-generated by MATLAB on 12-Sep-2007 15:29:06

% Create figure
figure1 = figure('Name','ALPINE (Version 3.1.25, May 2016)','Toolbar','none','Menubar','figure');
% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on');
box('on');
hold('all');
if angleConv~=1
    % Create multiple lines using matrix input to plot
    plot1 = eval(['mmpolar(' plotText(1:end-1) ')']);
    for k = 1:n
        set(plot1(k),'DisplayName',Label{k});
    end
    % Modify polar plot parameters
    mmpolar('RTickAngle',165,'RTickOffset',0.08,'TTickOffset',0.16,'TTickDelta',30);
    if exist('rhoMin','var'); mmpolar('RLimit',[rhoMin rhoMax]); end
    % Offset labels
    xlabelText = {' ';xlabelText};
    ylabelText = {ylabelText;' '};
    Title = {' ';Title;' ';' '};
else
    % Create multiple lines using matrix input to plot
    plot1 = eval(['plot(' plotText(1:end-1) ')']);
    for k = 1:n
        set(plot1(k),'DisplayName',Label{k});
    end
    % Set y axis range
    if exist('rhoMin','var'); axis([-1 1 rhoMin rhoMax]); axis('auto x'); end
end
% Create xlabel
xlabel(xlabelText);
% Create ylabel
if strcmp(varargin{1}.header.DATATYPE,'Antenna data')
    ylabelText = strrep(ylabelText,'RCS','Gain');
    if isempty(strfind(varargin{1}.header.FILENAME,'calibrated'))
        ylabelText = strrep(ylabelText,'dBsm','dB');
    else
        ylabelText = strrep(ylabelText,'dBsm','dBi');
    end
end
ylabel(ylabelText);
% Create title
title(Title,'FontSize',10,'FontWeight','bold');  % note: the blank offsets the label from the tick
…labels
```

```
256 % Create legend
257 legend1 = legend(axes1,'show');
258 set(legend1,'Location','SouthEast','Interpreter','None');
```

```
function plotRange(varargin)
% <<< Part of the ALPINE Code Suite >>>
% Revision: See "version.log" for revisions
% Version:  1.0  Original version, Dec 2007
% Copyright 2007, Peter J. Collins
% Air Force Institute of Technology
%
% plotRange(varargin)
%
% FUNCTION:
% Creates range resolution plot (1D)
%
% INPUTS:
% An arbitrary number of AFIT rcs or impulse response structures (optionally followed by the
…following)
% varargin = 'copol'                                to only plot co-pol data
%          = 'crpol'                                to only plot cross-pol data
%          = 'tt','pp','tp', or 'pt'                to plot particular pol
%          = 'caxis', [caxisMin caxisMax] (dBsm)    to set magnitude axis range
%          = 'raxis', [rngMin rngMax]               to set plot range in inches
%        If windowing is desired, choose one of the following strings...
%        'barthannwin', 'bartlett', 'blackman', 'blackmanharris', 'bohmanwin',
%        'chebwin' (windowParameter is max sidelobe level),
%        'flattopwin', 'gausswin' (parameter is 1/standard deviation),
%        'hamming', 'hann', 'kaiser' (parameter is beta parameter),
%        'nuttallwin', 'parzenwin', 'rectwin', 'triang',
%        'tukeywin' (parameter is flat to cosine taper ratio)
%
% OUTPUTS:
% none

% The AFIT data structure contains the following matrices...
%   out.frq (frequency vector in GHz)
%   out.ph (phi observation angle vector in degrees)
%   out.th (theta observation angle vector in degrees)
%   out.x (string system x axis position in meters)
%   out.y (string system y axis position in meters)
%   out.z (string system z axis position in meters)
%   out.roll (string system roll in degrees)
%   out.pitch (string system pitch in degrees)
%   out.yaw (string system yaw in degrees)
%   out.rng (down range distance vector in meters or time vector in seconds)
%   out.xrng (cross range distance vector in meters)
%   out.time (time vector in format YYMMDDHHMMSS)
%   out.tt (complex IQ phasor array for theta/theta polarization or TD data)
%   out.pp (complex IQ phasor array for phi/phi polarization)
%   out.tp (complex IQ phasor array for theta/phi polarization)
%   out.pt (complex IQ phasor array for phi/theta polarization)
%   out.header (structure containing file information)
% Note:  The out in dBsm is obtained by 20log10(abs(iq))
%        The phase in degrees is obtained by rad2deg(angle(iq))
%        Array format [freq <az> x angle <el>]

%% Add current directory to matlab's search path
currentDir = cd;
addpath(currentDir);
% Put any support *.m files in the resource directory
if exist(strcat(currentDir,'\resource'),'dir');
    addpath(strcat(currentDir,'\resource'));
    addpath(strcat(currentDir,'\resource\hipRCS'));
    addpath(strcat(currentDir,'\resource\facetPlot'));
end

%% Check for errors
if ~isstruct(varargin{1}) % Check for rcs structure
```

```
    disp('plotRange ERROR:  Input does not contain an RCS structure - aborting.  ')
    return
end

%% Initialize variables
c = 2.997925e8;  % speed of light (m/s)
n = 0;
X = {[]};
Y = {[]};
Z = {[]};
Label = {[]};
plotText2 = [];
copol = 0;
crpol = 0;
polStr = [];
windowType = 'rectwin';
windowParameter = [];
% Check for plot options
while ~isstruct(varargin{end})
    if isnumeric(varargin{end})
        if strcmp(varargin{end-1},'chebwin')  % Check for window type
            windowType = 'chebwin';
            windowParameter = varargin{end};
            varargin(end) =  [];  varargin(end) =  [];
        elseif strcmp(varargin{end-1},'gausswin')  % Check for window type
            windowType = 'gausswin';
            windowParameter = varargin{end};
            varargin(end) =  [];  varargin(end) =  [];
        elseif strcmp(varargin{end-1},'kaiser')  % Check for window type
            windowType = 'kaiser';
            windowParameter = varargin{end};
            varargin(end) =  [];  varargin(end) =  [];
        elseif strcmp(varargin{end-1},'tukeywin')  % Check for window type
            windowType = 'tukeywin';
            windowParameter = varargin{end};
            varargin(end) =  [];  varargin(end) =  [];
        elseif strcmp(varargin{end-1},'caxis')  % Check for color(magnitude) axis (dBsm) range
            caxMin = varargin{end}(1);
            caxMax = varargin{end}(2);
            varargin(end) = [];  varargin(end) =  [];
        elseif strcmp(varargin{end-1},'raxis')  % Check for plot range (inches)
            rngMin = varargin{end}(1);
            rngMax = varargin{end}(2);
            varargin(end) = [];  varargin(end) =  [];
        else
            disp('plotRange ERROR:  Numeric input provided without defining parameter - aborting.
')
            return
        end
    % Here for single part parameters
    elseif strcmp(varargin{end},'copol')  % Check if co-pol option desired
        copol = 1;
        varargin(end) = [];
    elseif strcmp(varargin{end},'crpol')  % Check if cross-pol option desired
        crpol = 1;
        varargin(end) = [];
    elseif strcmp(varargin{end},'tt')  % Check if tt-pol only option desired
        polStr = 'tt';
        varargin(end) = [];
    elseif strcmp(varargin{end},'pp')  % Check if pp-pol only option desired
        polStr = 'pp';
        varargin(end) = [];
    elseif strcmp(varargin{end},'tp')  % Check if tp-pol only option desired
        polStr = 'tp';
        varargin(end) = [];
```

```
129     elseif strcmp(varargin{end},'pt')  % Check if pt-pol only option desired
130         polStr = 'pt';
131         varargin(end) = [];
132     elseif strcmp(varargin{end},'barthannwin')  % Check for window type
133         windowType = 'barthannwin';
134         varargin(end) = [];
135     elseif strcmp(varargin{end},'bartlett')  % Check for window type
136         windowType = 'bartlett';
137         varargin(end) = [];
138     elseif strcmp(varargin{end},'blackman')  % Check for window type
139         windowType = 'blackman';
140         varargin(end) = [];
141     elseif strcmp(varargin{end},'blackmanharris')  % Check for window type
142         windowType = 'blackmanharris';
143         varargin(end) = [];
144     elseif strcmp(varargin{end},'bohmanwin')  % Check for window type
145         windowType = 'bohmanwin';
146         varargin(end) = [];
147     elseif strcmp(varargin{end},'flattopwin')  % Check for window type
148         windowType = 'flattopwin';
149         varargin(end) = [];
150     elseif strcmp(varargin{end},'hamming')  % Check for window type
151         windowType = 'hamming';
152         varargin(end) = [];
153     elseif strcmp(varargin{end},'hann')  % Check for window type
154         windowType = 'hann';
155         varargin(end) = [];
156     elseif strcmp(varargin{end},'nuttallwin')  % Check for window type
157         windowType = 'nuttallwin';
158         varargin(end) = [];
159     elseif strcmp(varargin{end},'parzenwin')  % Check for window type
160         windowType = 'parzenwin';
161         varargin(end) = [];
162     elseif strcmp(varargin{end},'rectwin')  % Check for window type
163         windowType = 'rectwin';
164         varargin(end) = [];
165     elseif strcmp(varargin{end},'triang')  % Check for window type
166         windowType = 'triang';
167         varargin(end) = [];
168     else
169         disp('plotRange ERROR:  Invalid plot parameter - aborting.  ')
170         return
171     end
172 end
173 
174 %% Loop through input structure building plot arrays
175 for k = 1:length(varargin)
176     % Check for errors
177     if ~isstruct(varargin{k})  % Check for RCS structure
178         disp('plotRange ERROR:  Arguement not an RCS structure  - aborting.  ')
179         return
180     elseif length(varargin{k}.ph)>1||length(varargin{k}.th)>1  % Check for more then one angle
181         disp('plotRange ERROR:  More than one angle, try plotGlobalRange() - aborting.  ')
182         return
183     end
184     % Build W, X, Y, Z, and Label matrices looping through polarizations
185     for pol = {'tt' 'pp' 'tp' 'pt'}
186         % Ignore if not co-pol or particular pol
187         if copol&&(strcmp(pol,'tp')||strcmp(pol,'pt')); continue; end
188         if crpol&&(strcmp(pol,'tt')||strcmp(pol,'pp')); continue; end
189         if ~isempty(polStr)&&~strcmp(pol,polStr); continue; end
190         % Ignore if no data
191         if isempty(varargin{k}.(char(pol))); continue; end
192         % Add polarization
193         n = n+1;
```

```
        % Calculate the range if not already calculated
        if isempty(varargin{k}.rng)
            tmp = calculateRange(varargin{k},windowType,windowParameter);
            Z(n) = {20*log10(abs(tmp.(char(pol))(:)))};
            X(n) = {tmp.rng(:)/0.0254};
            Label(n) = strcat(tmp.header.FILENAME,' (',pol,'-pol)');
        elseif strcmp(varargin{k}.header.DATATYPE,'IR data')  % Here for time domain input data
            Z(n) = {20*log10(abs(varargin{k}.(char(pol))(:)))};
            X(n) = {varargin{k}.rng(:)*c/2}; % range in inches
            Label(n) = strcat(varargin{k}.header.FILENAME,' (',pol,'-pol)');
        else  % Here for range input data
            Z(n) = {20*log10(abs(varargin{k}.(char(pol))(:)))};
            X(n) = {varargin{k}.rng(:)/0.0254};
            Label(n) = strcat(varargin{k}.header.FILENAME,' (',pol,'-pol)');
        end
        plotText2 = strcat(plotText2,'X{',num2str(n),'},Z{',num2str(n),'},');
    end
end
ph = num2str(varargin{end}.ph); if isempty(ph); ph = '0'; end
th = num2str(varargin{end}.th); if isempty(th); th = '0'; end
Title2 = ['Range Plot (\phi = ',ph,'\circ, \theta = ',th,'\circ)'];


%% Create plot (based on figure generated by MATLAB)
%CREATEFIGURE1(X1,YMATRIX1)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data

%  Auto-generated by MATLAB on 12-Sep-2007 15:29:06

% Create figure
figure2 = figure('Name','ALPINE (Version 3.1.25, May 2016))','Toolbar','none','Menubar','figure');
% Create axes
axes2 = axes('Parent',figure2,'YGrid','on','XGrid','on');
box('on');
hold('all');
% Create multiple lines using matrix input to plot
plot2 = eval(['plot(' plotText2(1:end-1) ')']);
for k = 1:n
    set(plot2(k),'DisplayName',Label{k});
end
% Create xlabel
xlabel('Range (inches)');
% Create ylabel
ylabel('Scattering (dBsm)');
% Create title
title(Title2,'FontSize',10);
% Create legend
legend2 = legend(axes2,'show');
set(legend2,'Location','SouthEast','Interpreter','None');
% Limit axes
if exist('rngMin','var')&&~exist('caxMin','var')
    axis([rngMin rngMax -1 1]);
    axis 'auto y';
elseif exist('caxMin','var')&&~exist('rngMin','var')
    axis([-1 1 caxMin caxMax]);
    axis 'auto x';
elseif exist('rngMin','var')&&exist('caxMin','var')
    axis([rngMin rngMax caxMin caxMax])
end
%
% if exist('rngMin','var')&&exist('rngMax','var'); axis([rngMin rngMax -1 1]); axis 'auto y'; end
% % Limit magnitude
% if exist('caxMin','var')&&exist('caxMax','var'); axis([rngMin rngMax -1 1]); axis 'auto y'; end

```