How to run the AR-HF Forward Modeling Code

1. Introduction

1.1. Purpose

The forward modeling code computes all possible two-dimensional shapes generated from a solid by either of two processes. One process projects a silhouette of the solid onto a plane, which is termed Plane of Projection. The other process, termed Plane of Section, examines the intersection of planes with the solid. For each possible two-dimensional shape, the following variables are computed: (1) aspect ratio (AR), (2) Heywood factor (HF), (3) solidity (S) and (4) concavity (C). These data are then used to determine the probabilities of any given pair of (AR, HF) value and any given (S,C) value being generated from a given solid.

1.2. Operational Overview

A solid is drawn using 3D modeling software and stored in objective file format (OFF). The MATLAB routine, *off2pds*, reads one or more OFF files in a directory. Using variables set in the command line the user specifies whether Plane of Projection or Plane of Section models are to be computed, the number of view points, maximum number of slices, and whether on not data formatted for gnuplot is to be output.

1.3. Algorithm Overview

off2pds() produces plane of projection and plane of section data. This is the only Matlab function that the user must interact with in order to run the forward modeling code. A high-level overview of the *off2pds()* logic follows.

1. Produce a triangulated mesh from the contents of an .off file.

2. Store the mesh and user inputs in a data structure.

3. Generate plane of section or plane of projection polygons.

4. Perform geometric measurements on these polygons.

4. Append the geometric parameters and polygons to the structure.

5. Write the data in the structure to .txt and .plt files. These files can be used to create PDS figures in gnuplot.

The following discussion details *off2pds()* and its primary dependencies in greater depth. It's necessary for the user to input, at minimum, a path to a directory containing one or more .off files, a number of sphere points (which is also the number of normal vectors), and a Boolean value that indicates whether plane of projection or plane of section data is to be generated. For plane of section, the number of sections per normal must also be provided. *off2pds()* will loop over all .off files in the given directory. If the name of a single .off file is provided, then *off2pd()s* will generate data for that file only. Other optional inputs include the *'OutputGnu'* flag, which tells *off2pds()* to create gnuplot-formatted .txt and .plt files, and the *'Pixelate'* flag, which tells *off2pds()* to pixelate all polygons. This simulates errors due to image binarization and region boundary identification

off2pds() passes the file name string of each .off file (in addition to some parameters that are specific to plane of projection or section) to *off2geomstruct()*. *off2geomstruct()* generates a data structure containing the triangle mesh generated from the .off data, the coordinates of all points distributed on the sphere, and the 3D convexity and solidity values for the mesh. The mesh is situated in the coordinate system such that its centroid lies at the origin. The centroid is found by calculating the mean of the coordinates of the mesh vertices.

Next, the plane of projection or section data is calculated for the solid by passing the newly created data structure to one of the two functions *planeofsection()* or *planeofprojection()*.

Note that if the *'OutputGnu'* flag was input by the user, *createGnuplotFiles()*, will be called when either of which generates the .plt and .txt files needed to generate AR-HF and C-S PDS figures.

planeofsection() calculates many cross-sections between the mesh and a plane. Aspect ratio, Heywood factor, convexity and solidity values are calculated for each of the resulting polygons. For each normal vector, N points are defined such that they are linearly spaced between the origin and the point that lies a distance $D/2$ projected along the current normal vector. N is the user-defined number of sections per normal vector and D is the Cartesian distance between the most distant pair of mesh vertices. Next, N unique planes are defined for each of the N points such that a plane contains one of the points and is orthogonal to the current normal vector.

planeofsection() uses *xsecmesh()* to produce the plane of section polygon(s) between a mesh and one plane. Holes (void spaces) are ignored. It was created for this application because no code could be found that was capable of reliably intersecting a plane with a concave mesh. This is due to the difficulty of geometric operations on concave geometries and to the presence of common mesh errors, such as more than two incident faces on one edge. It is important to note that the *xsecmesh()* output is empty when one or more mesh vertices lie in the cross-section plane.

Once all plane of section polygons have been calculated and stored in a cell array, *planeofsection()* loops over all of these polygons. Aspect ratio, Heywood factor, convexity and solidity are calculated for each polygon. If the *'Pixelate'* flag has been supplied to *off2pds()*, *particles8()* is used to simulate pixilation of the polygon before the geometric parameters are calculated. *planeofsection()* appends the cross-section polygons and the AR, HF, C and S cell arrays to the input data structure before returning the modified data structure.

planeofprojection() rotates the mesh to many different orientations. At each orientation, the triangular mesh faces are projected onto the XY plane. These projected faces form a set of co-planar polygons that are combined into a single, composite polygon. Holes (void spaces) are ignored. Aspect ratio, Heywood factor, convexity and solidity values are calculated for each for each polygon and stored in cell arrays which are appended to the input data structure before the structure is returned. The optional pixilation operations are identical to those implemented in *planeofsection()*.

The mesh is rotated using quaternion operations. Quaternions were because this method is faster than operations involving rotation matrices. This is also faster than varying the orientation of the plane with respect to a fixed mesh.

Projecting the mesh faces onto a plane produces a set of triangles. These triangular polygons must be joined in order to find the composite polygon. We use polygon Boolean *OR* operations (unions) to join the triangular polygons one-by-one until the final composite has been obtained. This is not a trivial task. Matlab's Mapping Toolbox includes *polybool*, a tool for polygon Boolean operations. We first used *polybool* but learned that it has a number of bugs. This is true for releases up to and including R2014b. Replacing *polybool* with Angus Johnson's *Clipper* resolved these issues.

It is worth noting that linear polygons are often problematic in polygon Boolean operations. Since projecting polygons onto a plane will sometimes produce a linear polygon, we have implemented a linear polygon check that throws away such polygons before the union procedure begins.

1.4. Theory

For a discussion of the theory and algorithm logic, see Rickman, Lohn-Wiley and Knicely, 2014, Probabilistic Particle Shape Measurement.

It should be noted that while Plane of Section and Plane of Projection polygons can contain holes (void space), this model discards polygons that bound void space and all polygons within void space.

1.4.1. Regarding pixilation

By default the AR, HF, C and S values computed by *off2pds()* are determined from double precision arithmetic on geometrically straight, line segments. This is a robust approximation of the ideal case. Actual measurements are made on rasterized data. With sufficient resolution there is little practical difference between the ideal and the rasterized data. However, in real measurements as particle size approaches the pixel size the difference between the ideal and the measured becomes significant. The *off2pds()* code allows examination of this effect. When the 'Pixelate' option is used polygons are converted to binary, rasterized approximations. These are then processed using the same logic as the PARTICLES8 class written by Gabriel Landini for ImageJ.

The pixilation process is controlled by the 'PixMaxArea' option. The 2D polygon with the largest area is found. From this polygon the maximum distance between two vertices is taken as a scaling factor. Assuming the polygon is square, the final scale factor is set such that after

rasterization the polygon would contain the number of pixels given by 'PixMaxArea'. The final scale factor is then applied to all polygons.

The rasterization process for each polygon, which uses the Matlab function poly2mask, is done after the polygon is translated along the x, y axes so that it is approximately centered in the mask.

1.5. Credits

This package incorporates work by Brian Hannan, Blake Lohn-Wiley, Joshua Knicely and Doug Rickman. The work was done over the summers of 2013 and 2014, as summer interns at NASA/Marshall Space flight Center, while under the direction of Doug Rickman. This code is an evolution of software written by Joshua Knicely and Blake Lohn-Wiley under the direction of Doug Rickman during the summer of 2013. Subsequently D. Rickman added capability to write files needed by gnuplot. The current code was written by Brian Hannan. Brian Hannan also created the logic used for pixilation and most of the logic for the concave solids.

1.6. Notices

This document and the associated software are actively evolving. Both were created for and maintained as part of a research effort conceived and led by Doug Rickman. A substantial amount of the coding has been done by Mr. Hannan, Lohn-Wiley and Knicely while they were unpaid volunteers. Ownership, license, warranty and all other aspects are within this framework.

2. Required Software

- MATLAB
- SketchUp (www.sketchup.com) or another CAD package that can export in OFF format
- the forward_modeling_code folder
- gnuplot (http://www.gnuplot.info) if plots are desired

3. Setup

Download the forward modeling folder and all of its subdirectories to your computer.

Once you have the forward modeling code on your machine, add this directory to the MATLAB search path. To do this launch MATLAB. Then go to File->Set Path...- >Add with Subfolders... and select the forward modeling folder. Select Open and then click Save.

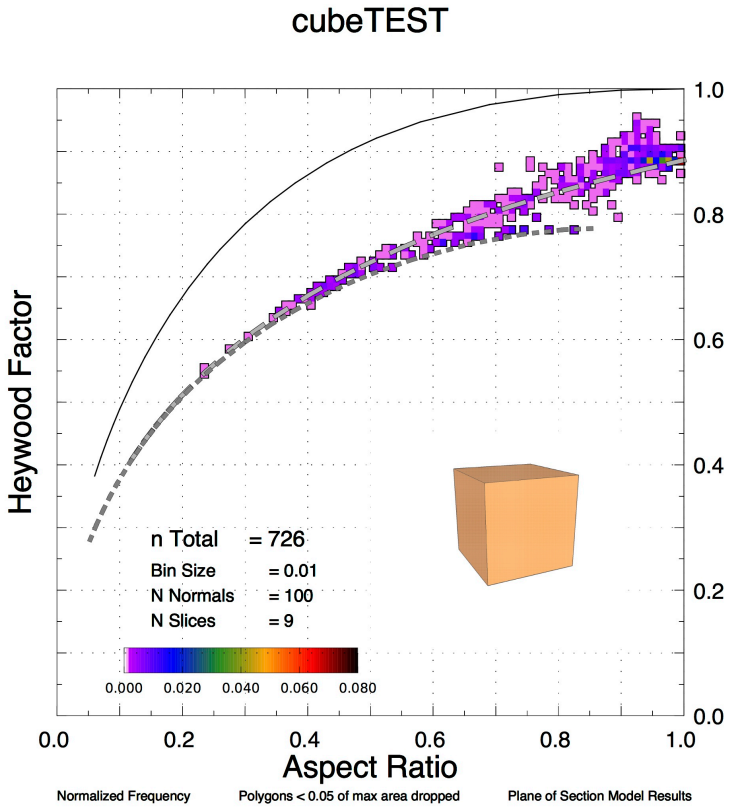
3.1. Testing

In the provided directory "Debugging" are the files:

- cubeTEST.skp
- cubeTEST.png
- cubeTEST.off
- cubeTEST_sec_100norms_9slices_PofSect.txt
- cubeTEST_sec_100norms_9slices_PofSect.plt
- cubeTEST_sec_100norms_9slices_PofSect.pdf

The .skp file was used to generate the .png and the .off files. The .off was the source file used to create a Plane of Section model with 100 normals and 9 slices. The output of the model was

the .plt and .txt files. These, with the .png file was used to make the following illustration. Note, to use the .plt file as is, it is necessary to place it, the .txt and the .png files in the directory indicated by the path1 and path2 statements at the beginning of the .plt file.



4. Using off2pds

4.1. User parameters

Within the MATLAB user environment, the user runs a routine called “off2pds” to create MAT file(s) containing AR-HF and C-S data for one or more OFF files. At least three input arguments are required for plane of projection. Plane of section requires a minimum of four input arguments.

Table 1. Required and Optional Inputs for off2pds.

Plane of Projection		**Plane of Section**	
Required	Optional	Required	Optional
dirStr	-	dirStr	-
secOrProj	-	secOrProj	-
nSphPts	-	nSphPts	-
-	-	nSlices	-
-	'FileName', fName	-	'FileName', fName
-	‘NumSigFig’, value	-	‘NumSigFig’, value
-	‘PixMaxArea’, value	-	‘PixMaxArea’, value
-	'OutputGnu'	-	'OutputGnu'

-	'Pixelate'	-	'Pixelate'

- ***dirStr*** A string (in single quotes) containing the full path to a directory. This must be the first input argument.
- ***secOrProj*** A case-insensitive string that indicates whether plane of projection or plane of section data is desired. This must be the second input argument.
  - Set ***secOrProj*** equal to 'proj' or 'projection' to generate plane of projection data.
  - When ***secOrProj*** equals 'sec' or 'section', plane of section data is generated.
- ***nSphPts*** The integer number of points to be distributed on the sphere (the number of normals). This must be the third input argument.
- ***nSlices*** The integer number of slices per normal. If plane of section data is desired, ***nSlices*** must be the fourth input argument. This input argument should only be supplied for plane of section.

Optional input arguments may be supplied in any order following the required arguments. Where single quotes are shown, single quotes must be entered at execution.

The optional input arguments in the form of name/value pairs are:

- ***'FileName', fName*** A property name-value pair input. When this name-value pair is provided, off2pds will generate AR-HF data only for the OFF file ***fName*** located in the directory ***directoryString***. ***fName*** is a string (in single quotes) that provides the name of the OFF file. ***fName*** must contain the -.off extension. The single quote marks are required.
- ***'NumSigFig', value*** An integer number for truncating vertex coordinates to the given number of significant digits. The default is 8.
- ***'PixMaxArea', value*** The approximate area, in pixels, of the largest polygon obtained from a solid. If the flag "Pixelate' is used and ***'PixMaxArea'*** is not set, the default of 15,000 will be used. See the "Theory" section for a discussion of what this parameter means.

The optional input arguments in the form of flags <u>are case sensitive</u>. These are:

- ***'OutputGnu'*** When the ***'OutputGnu'*** flag is supplied, PLT and TXT files will be generated for gnuplot figure generation. The single quote marks are required. Two PLT and two TXT files will be created. One pair contains concavity and solidity data and contains the string *_CS_* in the file name. The other pair contains aspect ratio and Heywood factor and contains *_ARHF_* in the file name.
- ***'Pixelate'*** When the ***'Pixelate'*** flag is supplied all polygons will be pixelated before AR and HF values are calculated.

For testing, run off2pds on a file in the testfolder directory (located inside forward_modeling_code/data_output_files/). To simplify the following examples a variable is first defined which contains the value of ***directoryString***.

mydir = '/[YOUR PATH HERE]/data_output_files/testfolder/';

The following command will run plane of section code with 101 normals and 22 sections per normal for all OFF files in the directory specified in mydir.

off2pds(mydir,'sec',101,22);

The following command will run plane of projection code for all OFF files in the directory specified in mydir.

off2pds(mydir,'proj',1000);

The following command will run plane of section code with 1500 normals and 13 sections per normal only for the OFF file named block.off in the directory specified in mydir.

off2pds(mydir,'sec',1500,13,'FileName','block.off');

The command below tells off2pds to generate plane of section data and to additionally generate the PLT and TXT files required by gnuplot.

off2pds(mydir,'sec',100,10,'FileName','block.off','OutputGnu');

To do plane of section models for all OFF files in *mydir* with 9 significant digits for vertices do the following.

off2pds(mydir,'sec',100,10,'NumSigFig',9);

To do plane of projection models for all OFF files in *mydir* and pixelate such that the larges polygon is approximately 1000 pixels do the following.

off2pds(mydir,'proj',1000,'Pixelate','PixMaxArea',800)

Once off2pds has generated AR-HF and C-S data, it will bin the data in preparation for plotting the PDS. The bin interval constant *BIN_INTERVAL* is defined at line 99 of off2pds.m, where its value is set to 0.01.

Immediately below *BIN_INTERVAL* is a line that defines the constant *MIN_FRACTION*. All polygons having an area less than *MIN_FRACTION * [maximum polygon area]* are dropped. By default, *MIN_FRACTION* is set to 0.05, and therefore polygons are ignored if their area is less than 5% of the maximum polygon area.

4.2. Internal parameters

In the off2pds.m source code there are three parameters which may be changed if one has access to the code. Their names and default values are:

BIN_INTERVAL = 0.01
MIN_FRACTION = 0.05
UPDATE_FRACT = 0.2

BIN_INTERVAL determines the size of bins used for computing the frequency of occurrence of AR, HF, C and S.

MIN_FRACTION limits the retention of small polygons in the models. The polygon with the largest area, maxArea, is found. Polygons smaller than MIN_FRACTION*maxArea are discarded.

UPDATE_FRACT is a percentage of the total processing. Off2pds prints a status message each UPDATE_FRACT.

4.3. .mat File Output

off2pds.m generates a .mat file each time it runs. This .mat file contains a single data structure that holds mesh data and plane of projection or plane of section data and is saved to the directory that is supplied as off2pds's first argument.

The data structure in the .mat file is named GeomStruct. The fields of GeomStruct are:

- **offDirectory** The directory that contains the .off file (a string).
- **offFileName** The name of the .off file (a string).
- **shapeName** A string used to generate the PDS plot title and the file name for the PDS .pdf files.
- **isSec** A Boolean value that is set to true for plane of section data.
- **nSphPts** The integer number of points distributed on the sphere.
- **cSteps** (section only) The number of "slices" per normal.
- **cVals** (section only) A row vector containing the radius values for each of the "slices." Thus cVals has a length equal to cSteps + 1.
- **verts** The vertices matrix for the mesh.
- **faces** The faces matrix for the mesh.
- **edges** The edges matrix for the mesh.
- **minVertDist** The Cartesian distance between the minimally-separated mesh vertices.
- **xs** A row vector containing the x coordinates of all points distributed on the sphere.
- **ys** A row vector containing the y coordinates of all points distributed on the sphere.
- **zs** A row vector containing the z coordinates of all points distributed on the sphere.
- **isPix** A Boolean value that is set to true if the polygons in the file have been pixelated.
- **pixMaxArea** A numerical value used for scaling pixelated polygons. This variable will be present and non-zero even if the polygons in the file were not pixelated.
- **convy3d** The 3D convexity value for the mesh.
- **soly3d** The 3D solidity value for the mesh.
- **matFileName** A string that provides the name of the .mat file without the ".mat" extension. This string is stored within the .mat file because it simplifies the .pdf file naming process (see writeGnuFile.m).
- **runTime** The run time in seconds.
- **hf** A cell array containing Heywood factor data. The cell array has 1 row and nSphPts columns. Each element in this cell array is itself a cell array. Each of these "sub-cells" contains all Heywood factor values generated in one plane of section/projection. hf exists as a cell of cells so that all values generated from the same section/projection can be stored together.
- **ar** A cell array containing aspect ratio data. The cell array has 1 row and nSphPts columns. Each element in this cell array is itself a cell array. Each of these "sub-cells" contains all aspect ratio values generated in one plane of section/projection.

- **convy** A cell array containing concavity data. The cell array has 1 row and nSphPts columns. Each element in this cell array is itself a cell array. Each of these "sub-cells" contains all concavity values generated in one plane of section/projection.
- **soly** A cell array containing solidity data. The cell array has 1 row and nSphPts columns. Each element in this cell array is itself a cell array. Each of these "sub-cells" contains all solidity values generated in one plane of section/projection.
- **solidVerts** (projection only) A cell array containing the vertices of the mesh at each orientation. The cell array has 1 row and nSphPts columns. Each element is a matrix that has the same size as the vertices matrix.
- **prjVts** (projection only) A cell array containing the vertices of the plane of projection polygon at each orientation. The cell array has 1 row and nSphPts columns. Each element is an Nx2 matrix.
- **isProjPolygonInvalid** Since polybool.m has bugs which sometimes unpredictably cause the projection union process to fail, I have used a try/catch statement that allows the code to continue operating when a bug in polybool causes an error. isProjPolygonInvalid is a cell array containing Boolean values that indicate whether the plane of projection union process failed at each solid orientation. This cell array has 1 row and nSphPts columns.
- **tfInt** (section only) A logical row vector. tfInt has a length equal to numSphPts*cSteps. A false entry indicates that the cutting plane and solid did not intersect at that orientation.
- **planes** (section only) A cell array containing each cutting plane, stored as a 1x9 array. The 1st 3 elements are a point on the plane. The following two groups of 3 elements are two unique in-plane vectors.
- **isPlSecInvalid** (section only) A logical row vector that is true for plane/mesh intersections that failed to generate a plane of section polygon due to the intersection of the plane with a mesh vertex.

5. Using mat2pds

mat2pds.m (in the gnuplot_functions folder) can be used to create convexity-solidity gnuplot files from existing .mat files. The function has only one input argument:

- ***directoryString*** A string (in single quotes) containing the full path to a directory. This must be the first input argument.

```
inDir = '/path/to/a/directory';
mat2pds(inDir);
```

mat2pds.m accepts a single string as an input. The string must contain a path to a directory containing .mat files that have been created by off2pds.m. The function iterates over all .mat files in the directory, creating gnuplot .txt and .plt files from each. If C-S data already exists in the .mat file, the gnuplot files are created immediately. If not, the C-S values are calculated from the polygons that exist in the .mat file before creating the gnuplot files. The new .txt and .plt files are output to the directory that is passed to mat2pds.

6. Creating Solids

Many 3D design packages can be used to define solids. The following discusses doing this with SketchUp.

6.1. Exporting SketchUp drawings in OFF format

After installing SketchUp, download the SketchUp plugin, off_exporter.rb. This may be downloaded from http://www.cs.princeton.edu/courses/archive/spr08/cos426/assn2/off_exporter.rb Place off_exporter.rb in the SketchUp 2014->SketchUp->Plugins folder. More info on locating this folder can be found at http://help.sketchup.com/en/article/38583. To export a SketchUp drawing, select Plugins->Export Off.

6.2. Drawing the Solid

The forward modeling code requires that the mesh's diameter be large compared to E^{-6} units and small compared to E^{+18} units. We recommend that the mesh has a diameter of ~1 units. This requirement is satisfied when the SketchUp drawing's maximum diameter is on the order of one meter.

The forward modeling code requires that the solid be represented by a closed surface and assumes there are no elements other than those needed to form the exterior surface of the solid. Thus, extra lines or surfaces can not be present in the SketchUp drawing when the .off file is created. A particular problem to be aware of are "interior faces" An interior face is a polygon that passes through the volume enclosed by the boundary of the closed surface. Even without the user intentionally creating interior faces, SketchUp sometimes automatically forms interior faces in a solid. This will cause an error in the plane of section code because a non-ideal (non-manifold) triangle mesh will be formed during import. Specifically, ideal meshes have at most 2 incident faces per edge). It is therefore necessary to check for interior faces before exporting the SketchUp drawing.

Figure 1 shows a beveled cube that was drawn in SketchUp. This is contained in the file Cube_Beveled Initial.skp. Figure 2 shows the solid with several of the exterior faces hidden, so internal structure may be seen. Selected faces may be hidden by right-clicking and using the Hide option. In Figure 2, multiple interior faces may be seen. Figure 3 shows the view after the internal faces were removed. After deleting any interior faces, the drawing can be exported using the OFF format. To unhide a hidden face, turn on the "Hidden Geometry" option in the view menu. The hidden face may be then be selected and a right click will show the Unhide option.

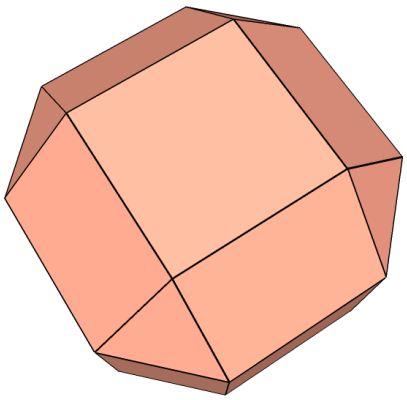


Figure 1: Initial SketchUp drawing showing exterior faces only.

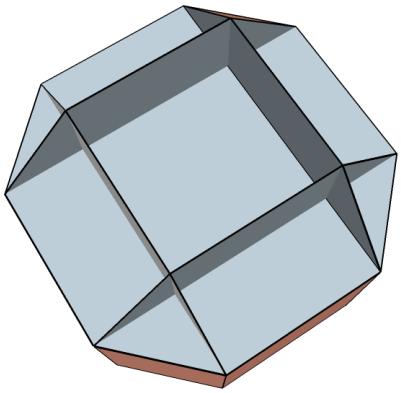


Figure 2. View after hiding several exterior faces. Note the multiple interior planes.

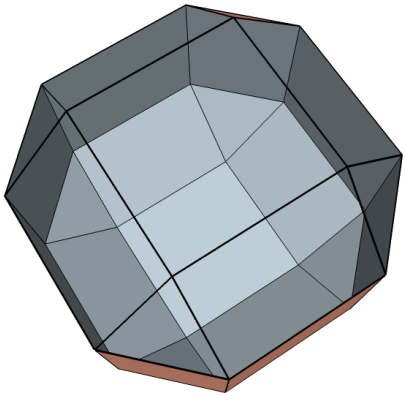


Figure 3. Same view as in Fig. 2 after removal of interior faces.

6.3. Image file

It can be useful to have a reference image showing the solid. This may be made in SketchUp using the File – Export - 2D Graphic menu. Export should be as a .png file. Under options for the .png set the output file width to 1000. This will make the image size suitable for inclusion in a plot made by gnuplot, as shown in Fig. 4.

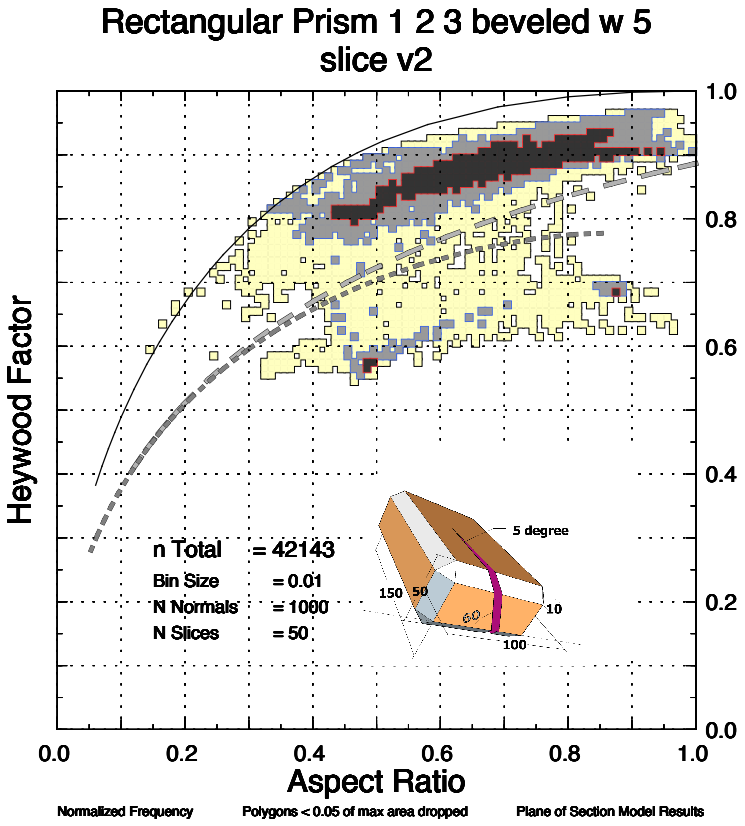


Figure 4. Example plot showing embedded image of the solid.

To make an image that communicates the geometry of the solid can require some care. There are several things that will help considerably in effective visualization of the solid:

- color coding groups of facets,
- show critical dimensions,
- how edges are shown,
- use of a shadow.

As examples, see Fig. 5 and 6.

Dimensions added in SketchUp are not exported into the OFF file. In order to have the dimensions clearly legible in the PDF generated by gnuplot, the point size of the dimension text needs to be relatively large. This is set from the menu: Window - Model Info - Dimensions. A bold font with a size of 36 – 50 points generally works well.

Also, text added in SketchUp is not exported into the OFF file. Window - Model Info – Text allows setting the point size of the text. Parameters for text with a leader line and text without a leader line are set separately. A bold font with a size of 36 – 50 points generally works well.

It is sometimes useful to add lines or curves to the illustration. For example in Fig. 6 there are several lines which help visualize the angular rotation of 30°. Such elements can not be present in the drawing when the OFF file is created.

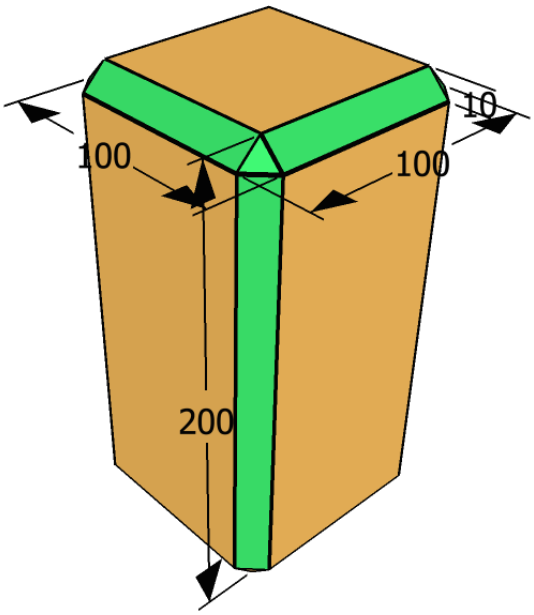


Figure 5. PNG image of model solid showing critical dimensions and color-coded facets.

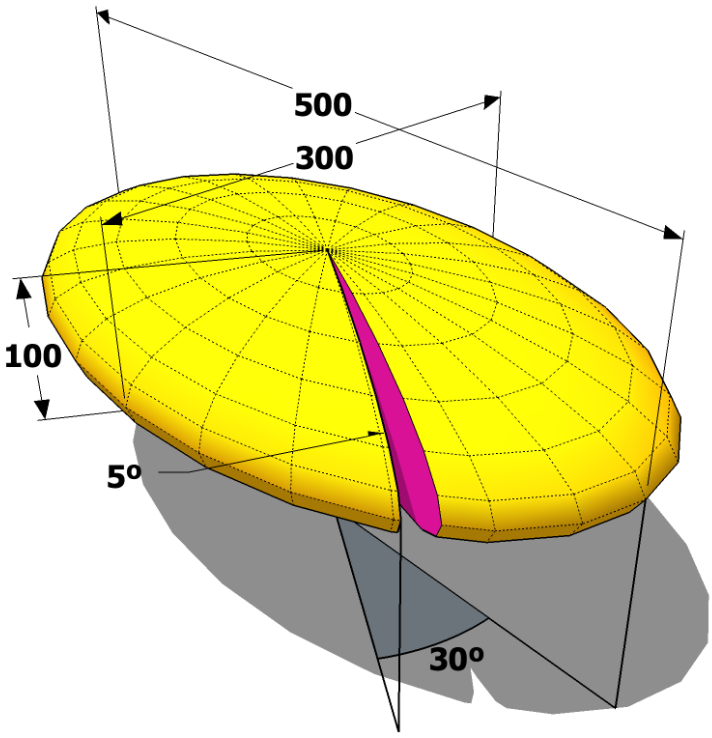


Figure 6. PNG image of a model solid using shadowing. The "5°" text is an example of text with a leader line. The "30°" text is an example of text without a leader line. The arc line adjacent to the "30°" text, the lines in the plane of shadow defining the 30°, and the drop lines from the perimeter were added after the .off file was created.

6.4. OFF file format

Objective File Format files begin with the keyword OFF. The next line holds three integers: the number of vertices, the number of edges, and the number of faces. A list of the Cartesian coordinates of all vertices follows (one per line). Next, faces are listed (one per line). For a triangle mesh, a given face will be listed as three indices that point to the three vertices that form the triangular face.

6.5. Design Considerations for Solid

The algorithm used in the forward modeling code breaks a solid into triangular facets. The total number of facets needed to represent the solid is a major driver of the run time for a model. Therefore, when creating a solid it is prudent to minimize the number of facets used. One way to minimize the number of facets is to use unrounded edges. However, solids with unrounded edges will generate intersections that are triangular in outline. This can result in unrealistic populations along the triangle line. But rounded edges are rendered in SketchUp as segmented lines, which increases execution time. Thus, there is commonly a conflict between needing to round edges to avoid generation of unwanted triangular intersections while not creating an excessive computational burden.

Rounding can be effectively approximated by use of bevels. However, drawing bevels in SketchUp is difficult. A utility to do this is available and has been shown to be satisfactory for our work. It is call "Round Corner," version 2.6a, 27 Nov 13. It is copyrighted by Fredo6 - © June 2009. The author's contact information is

Didier Bur
CRAI-ENSAN
2 rue Bastien-Lepage
BP 40435
54001 Nancy Cédex – FRANCE
bur@crai.archi.fr

7. Regarding distributing points on the sphere

There are two different routines for distributing points evenly over the surface of a sphere. One places the points along a spiral and is based on an algorithm published in "Distributing many points on a sphere" by E.B. Saff and A.B.J. Kuijlaars, Mathematical Intelligencer 19.1 (1997) 5--11. This was rendered as code by Dirk Laurie (dlaurie@na-net.ornl.gov) in newsgroup sci.math.num-analysis 24 Apr 1997. The distances between all points, except the end points of the spiral, are equal. A later, improved implementation is found in Bauer, Robert, "Distribution of Points on a Sphere with Application to Star Catalogs ", Journal of Guidance, Control, and Dynamics, January-February 2000, vol.23 no.1 (130-137). For more information see http://sitemason.vanderbilt.edu/page/hmbADS#code. Another algorithm was created by Cheng Guan Koay, cgkoay@wisc.edu. This algorithm is claimed to be an exactly uniform distribution of points over the sphere. See Koay, Cheng Guan. 2011. "Analytically Exact Spiral Scheme for Generating Uniformly Distributed Points on the Unit Sphere." *Journal of Computational Science* 2 (1) (March 1): 88–91. doi:10.1016/j.jocs.2010.12.003.

Implementation notes - PointsOnSphereEqually throws an error the first time (and only the first time) it runs in a new MATLAB session. The function requires the creation of a java object to run, and it doesn't seem to want to do this on the first try. Brian has temporarily stopped using it to avoid problems with automated data generation.

Brian commented the call to PointsOnSphereEqually at lines 97-98 and replaced it with PointsOnSphere at lines 99-100 until the error can be avoided. If you would like to use PointsOnSphereEqually, modify off2geomstruct so that it no longer calls PointsOnSphere and run PointsOnSphereEqually once before generating your AR- HF data.

Inside PointsOnSphereEqually, a path is set that points to the location of the JAR file. Be sure to modify this path before use. The line that requires modification is line 46 of PointsOnSphereEqually.m (in the common_code folder).

8. External Code

1.1. matGeom

This code uses some functions from the matGeom package version 1.1.9 (released Feb. 2014). Visit the matGeom page, http://matgeom.sourceforge.net/wiki/index.php/MatGeom

The matGeom functions used are

- centroid.m
- createLine3d.m
- createPlane.m
- distanePointLine3d.m
- distancePointPlane.m
- intersectEdgePlane.m
- meshEdges.m
- normalizeVector3d.m
- readMesh_off.m
- vectorCross3d.m
- vectorNorm3d.m

1.2. Clipper

In order to generate plane of projection polygons, we perform unions (polygon OR operations) using version 6 of Angus Johnson's Clipper library (http://www.angusj.com/delphi/clipper.php). This library is called from the MATLAB environment with Emmit's wrapper, which can be found on the MathWorks file exchange (http://www.mathworks.com/matlabcentral/fileexchange/36241-polygon-clipping-and-offsetting).

1.3. unifyMeshNormals

We make use of the function unifyMeshNormals.m (www.mathworks.com/matlabcentral/fileexchange/43013-unifymeshnormals).

1.4. inPolyhedron

We use Sven's inPolyhedron.m (http://www.mathworks.com/matlabcentral/fileexchange/37856-inpolyhedron-are-points-inside-a-triangulated-volume-).

9. Running gnuplot

In order to produce high quality graphics gnuplot, version 4.6, is used. For this reason off2pds writes two types of ASCII files, one with data and the other with a command script to run gnuplot. By running the script from the command line, gnuplot will produce a PDF of the plot. The names of all three files are determined by the name of the OFF file with the extension ".txt", ".plt", and ".pdf" respectively.

off2pds.m generates two sets of paired ".txt" and ".plt" files. One pair is formed from the aspect ratio and Heywood factor data. The other pair contains the convexity and solidity data. PLT and TXT files are created for each data set. Therefore, *off2pds.m* outputs two PLT files and two TXT files. The files that correspond to the aspect ratio and Heywood factor data contain the string "*_ARHF_*" in the file name. The files that correspond to the concavity and solidity data contain the string "*_CS_*" in the file name.

The AR-HF PLT and TXT files contain code that will plot the ellipse, triangle and rectangle lines on the PDS figure. These commands are not included in the C-S files and thus the ellipse, triangle and rectangle lines are not plotted on concavity-solidity figures.

9.1. .TXT file

In the .txt file there are comment lines, defined by a leading "#", containing information about the data. Then there are six sets of data. The following is an example of the comment lines for a dipyramidal trigonal prism.

```
# Plane of Section Model Results  -  Formatted for gnuPLOT
# Corresponding gnuPLOT script file =
   "Dipyramidal_Trigonal_Prism_1_1_PofSect.plt"
#
# Dipyramidal Trigonal Prism 1 1
# Model execution time = 19.8475 seconds
# Number of Surface Normals : 100
# Slice Spacing : 10
# Polygons smaller than 0.05 of the largest polygon are dropped.
# Number of data points = 639
# Binning Interval = 0.01
# Max frequency in a bin                      = 0.025039
# Frequency at which cumulative ~50% occurs   = 0.0031299
# Frequency at which cumulative ~75% occurs   = 0.0015649
# Frequency at which cumulative ~95% occurs   = 0.0015649
# Cum Freq for all bins with 5 or more points = 0.10016
#
# Cumulative Frequency within frequency 0.0015649, which is contour 1 =
   0.95149
# Cumulative Frequency within frequency 0.0062598, which is contour 2 =
   0.16745
# Cumulative Frequency within frequency 0.010955, which is contour 3 =
   0.034429
```

```
# Cumulative Frequency within frequency 0.015649, which is contour 4 =
   0.034429
# Cumulative Frequency within frequency 0.020344, which is contour 5 =
   0.034429
#
# Data Block 1: Ellipse, rectangle and triangle lines
# Data Block 2: Binned and normalized data, structured for contouring
# Data Block 3: Binned and normalized data with cumulative frequency >= 95%
# Data Block 4: Binned and normalized data with cumulative frequency >= 75%
# Data Block 5: Binned and normalized data with cumulative frequency >= 50%
# Data Block 6: Individual Aspect Ratio and Heywood points.
```

9.2. .PLT file

The details of the .plt file are beyond the scope of this document. An example .plt is included here with a few additional comments. For syntax and functional significance of individual commands please see the gnuplot documentation.

Please note, the .plt file continues to evolve as we develop our work. This version may or may not match exactly with the .plt generated by your installation.

Comment – In a gnuplot script, where a "#" is found defines the start of a comment. The comment continues to the end of the line.

```
# - - - - - Introduction - - -  - - - - - - - - - - - - - - - - - -
# Plane of Section Model ResultsFor the solid Dipyramidal Trigonal Prism 1 1
# Graph Relative Frequency of Aspect Ratio vs Heywood Factor using gnuplot.
# The path is set to /Users/doug2/Documents/MATLAB/.  Edit the "path1" and
   "path2 lines to change this.
# This script designed by Doug Rickman, Jan 30, 2014, mod: July 3, 2014
# set terminal pdf enhanced dashed dl 3 size 7,7
set terminal pdfcairo enhanced color dashed dashlength 3 size 7,7   # The
   size affects the dimensions of the plot.
```

Comment – Macros are used to simplify both manual editing of the script and its original creation. This way file names and paths can be defined at the front of the script. Then actual commands can refer to the definitions made using the macros.

```
# - - - - - Define Macro Terms. Set Input and Output Files - -
set macros
dquote    = '"'
path1     = "/Users/doug2/Documents/"
path2     = "MATLAB/"
outfile   = "Dipyramidal_Trigonal_Prism_1_1_PofSect.pdf"
infile    = "Dipyramidal_Trigonal_Prism_1_1_PofSect.txt"
set output @dquote@path1@path2@outfile@dquote

# - - - - - Define Line Styles -- - - - - - - - - - - - - - - - - -
set style line 1  lt 0  lw 3  lc rgbcolor 'black'   # dot
set style line 2  lt 1  lw 3  lc rgbcolor 'black'   # solid
set style line 3  lt 2  lw 10 lc rgbcolor 'grey70'  # long dash
set style line 4  lt 3  lw 10 lc rgbcolor 'grey50'  # short dash
set style line 5  lt 4  lw 3  lc rgbcolor 'black'   # long short long short
```

```
set style line 6  lt 5  lw 3  lc rgbcolor 'black'   # long short short long
set style line 13 lt 2  lw 12 lc rgbcolor 'black'   # long dash
set style line 14 lt 3  lw 11 lc rgbcolor 'black'   # short dash
```

Comment – The plots are actually 3D surfaces, though they are not shown in a way that this is obvious. The general controls configure various parameters that may be used by subsequent plot operations.

```
# - - - - - Set General Controls - - - - - - - - - - - - - - - -
set grid                        # grid and cntrparam are used with contours
set cntrparam bspline
set cntrparam order 10
set view equal xyz              # View establishes the orientation and visual
  geometry of the plot
set view 0,0,1.5,1              # This affects the dimensions of the plot.
unset xlabel                    # Remove any existing X, Y, Z axes labels
unset ylabel
unset zlabel
unset key                       # Do not display a key
unset zrange                    # Free the range of Z values
unset colorbox                  # Turn off the color box
set xrange [0:1]                # Set the X range of the plot
set yrange [0:1]                # Set the Y range of the plot
set zrange [0:100]              # Set the Z range of the plot
unset ztics                     # Turn off tics on the Z axis
set mxtics 2                    # Define minor tics frequency on X axis
set mytics 2                    # Define minor tics frequency on Y axis
set multiplot                   # Allow multiple plots in the same space
set size 0.95, 0.95             # Set the size of the plot
set origin 0.0, 0.01            # Set the starting location of the plot
```

Comment – The labels are automatically generated in off2pds based on the user defined options.

```
# - - - - - Define the Labels - - - - - - - - - - - - - - - - - -
set label 1 'Dipyramidal Trigonal Prism 1 1'            font 'Helvetica,30'
  at  0.5 ,  1.1 , 0 center
set label 2 'n Total   {\011}= 639'  font 'Helvetica,20' at  0.15,  0.23 , 0
set label 3 'Bin Size  {\011}= 0.01' font 'Helvetica,17' at  0.15,  0.18
set label 4 'N Normals{\011}= 100'   font 'Helvetica,17' at  0.15,  0.14
set label 5 'N Slices  {\011}= 10'   font 'Helvetica,17' at  0.15,  0.10
set label 6 'Aspect Ratio'           font 'Helvetica,28' at  0.5 ,  -0.09, 0
  center
set label 7 'Heywood Factor'         font 'Helvetica,28' at -0.05 ,  0.5 , 0
  center rotate by 90
set label 8 'Normalized Frequency'   font 'Helvetica,12' at screen 0.10,
  0.02, 0 left
set label 9 'Polygons < 0.05 of max area dropped' font 'Helvetica,12' at
  screen 0.45,  0.02, 0 center
set label 10 'Plane of Section Model Results' font 'Helvetica,12' at screen
  0.64,  0.02, 0 left

# - Create grid lines and tics every 0.2 intervals on both axes -
set xtics 0.2 format "%.1f" font 'Helvetica,20' offset -0.35,-0.3
set ytics 0.2 format "%.1f" font 'Helvetica,20' offset -0.35,-0.3

set surface                     # Show the data as a surface
unset contour                   # Do not show the data as contours
```

Comment – The .plt file can be used to make a large number of different plots. Which plots are produced is controlled by uncommenting and commenting the following lines.

```
# - - - - - Choose Which Data To Plot And How - - - - - - - - - - -
# - Later Plots Overwrite Earlier Plots BUT DO NOT ERASE THEM - -
# - -  - Plots may be reordered by changing their sequence. - - -

# To plot the individual data points turn on the following line.
# splot @dquote@path1@path2@infile@dquote  index 5 using 1:2:3 with points
   pointtype 7 lc rgbcolor 'dark-green'  pointsize 0.2 # filled circles

# In the 95, 75 and 50% plots, the data are plotted twice, to create an
   outline effect.
# The bins are plotted as points using a hollow square symbol!!!
# The point size of each of a pair of plots is scaled to match the bin size
   used and the dimensions of the plot.
# The first plot of a pair is larger and creates an black outline effect for
   the data cloud.
# The plotted symbols are shifted so they align graphically on the bins in
   the plot.
# If something changes the dimensions of the plot, the size of the symbols
   must be changed.
```

Comment – The data produced by MATLAB are actually bin boundaries. Gnuplot is using the boundaries as though they are points. Therefore, it is necessary to translate the MATLAB values to the center of the bin as plotted by gnuplot. This is done by shifting the MATLAB data one half the width of the bin, which in this case is 0.005.

```
# To plot the bins containing 95% of the data.
splot @dquote@path1@path2@infile@dquote  index 2 using
   ($1+0.005):($2+0.005):($3) with points pointtype 5 lc rgbcolor 'black'
   pointsize 0.8    # Hollow squares, used to outline
splot @dquote@path1@path2@infile@dquote  index 2 using
   ($1+0.005):($2+0.005):($3) with points pointtype 5 lc rgbcolor
   'lemonchiffon'  pointsize 0.60    # solid squares

# To plot the bins containing 75% of the data
splot @dquote@path1@path2@infile@dquote  index 3 using
   ($1+0.005):($2+0.005):($3) with points pointtype 5 lc rgbcolor 'royalblue'
   pointsize 0.78 # solid squares
splot @dquote@path1@path2@infile@dquote  index 3 using
   ($1+0.005):($2+0.005):($3) with points pointtype 5 lc rgbcolor 'grey60'
   pointsize 0.60 # solid squares

# To plot the bins containing 50% of the data
splot @dquote@path1@path2@infile@dquote  index 4 using
   ($1+0.005):($2+0.005):($3) with points pointtype 5 lc rgbcolor 'light-red'
   pointsize 0.78 # solid squares
splot @dquote@path1@path2@infile@dquote  index 4 using
   ($1+0.005):($2+0.005):($3) with points pointtype 5 lc rgbcolor 'grey20'
   pointsize 0.60 # solid squares

# To plot the binned data as contours turn on the following 9 lines.
# unset surface
# set contour
# set cntrparam levels incremental 0.00156, 0.00469, 0.025
```

```
# set cbrange [0.0:0.025039]
# set label 11 'Contour (B,I,E)= 0.00156, 0.00469, 0.025'   font
   'Helvetica,17' at  0.15,  0.06
# splot @dquote@path1@path2@infile@dquote  index 1 using
   ($1+0.005):($2+0.005):($3) w l lw 3 palette
# set surface
# unset contour
```

Comment – In the following, the color assigned to a bin is controlled by the value of the bin, the palette, and the scaling from bin values to the palette range. The scaling is done in the "set cbrange" command. The values used by cbrange default from zero to the maximum observed value for the data set being plotted. It can be useful to change the maximum to a slightly larger, evenly divisible value and then change the value of cbtics to a value that is exactly 1/2 to 1/4 of the new maximum. For this example, the value of 0.025039 might be changed to 0.026 and the 0.002 changed to 0.013.

```
# To plot the normalized, binned data as a greyscale or a colored surface
   first turn on 1 of the following 2 lines.
# set palette defined (  0 "white", 0.1 "grey", 5 "black")   # Use this line
   for a greyscale plot.
# set palette defined (  0 "white", 0.1 "light-magenta", 0.1 "dark-magenta",
   1 "blue", 2 "forest-green", 3 "orange", 4 "red", 5 "black") # Use this line
   for colored plot.
# Second, turn on the following 5 lines.
# set colorbox horizontal user origin 0.18,0.17 size 0.28,0.03
# set cbrange [0.0:0.025039]
# set cbtics 0.002 format "%.3f" # Adjust the tic interval from 0.002 as
   needed
# splot @dquote@path1@path2@infile@dquote  index 1 using
   ($1+0.005):($2+0.005):($3>0.00000 ? $3 : NaN) with points pointtype 5 lc
   rgbcolor "black"  pointsize 0.78
# splot @dquote@path1@path2@infile@dquote  index 1 using
   ($1+0.005):($2+0.005):($3>0.00000 ? $3 : NaN) with points pointtype 5
   palette  pointsize 0.68

# - - - - - Do the Ellipse, Rectange and Triangle Lines - -  - -
# To plot the lines for ellipses, rectangles and triangles turn on the
   following lines.
set format xy""
set xtics 0.1
set ytics 0.1
splot @dquote@path1@path2@infile@dquote   index 0 using 1:2:7 w l ls 2  #
   Ellipse
splot @dquote@path1@path2@infile@dquote   index 0 using 3:4:7 w l ls 13 #
   Rectangle
splot @dquote@path1@path2@infile@dquote   index 0 using 3:4:7 w l ls 3  #
   Rectangle
splot @dquote@path1@path2@infile@dquote   index 0 using 5:6:7 w l ls 14 #
   Isoceles triangle
splot @dquote@path1@path2@infile@dquote   index 0 using 5:6:7 w l ls 4  #
   Isoceles triangle
unset surface
set contour
```

Comment – The following will place a picture of the solid, saved in the file "my_solid.png", in the lower right quadrant of the plot. The picture may be generated in SketchUp. The following parameters work well if the .png file is approximately 1000 pixels wide.

```
set style rectangle back fc rgb "white" fs solid border
set object 101 rectangle from screen 0.45, 0.2, 0 to screen 0.75, 0.45, 0
set cbrange [*:*]
splot "my_solid.png" binary filetype=auto origin=(0.44,0.1,0) dx=0.0005
  dy=0.0005 with rgbimage

unset multiplot                  # This causes all of the plots defined above
  to be rendered into the output file.
exit
```

9.3. Installation of gnuPLOT

It is necessary to have gnuplot properly installed and with the correct options. The script in the .plt file uses the terminal "pdfcairo" and employees the "enhanced" option in order to generate dashed lines. This option must be selected during the installation process.

On the Mac, under OS X 9.4, we have found it easiest to install gnuplot using MacPorts, https://www.macports.org/ports.php?by=name&substr=gnuplot The variants used are pangocairo and pdflib

Further guidance about installing gnuplot is beyond the scope of this document.

10. Information

For additional information please contact

Doug Rickman
Earth Science Office MSFC/NASA
320 Sparkman Drive
Huntsville, Alabama 35805

256-961-7889
doug.rickman@nasa.gov

11. Dependencies for off2pds

off2pds contains 59 separate matlab routines. The structure is complex. It uses two matlab toolboxes: Image Processing, and Statistics. It also uses the 3rd party Clipper package, as noted previously. To understand the relationships between the routines we recommend use of a package such as fdep, which may be obtained from http://www.mathworks.com/matlabcentral/fileexchange/17291-fdep--a-pedestrian-function-dependencies-finder An example of how to run this is:

fdep('/PATH/forward_modeling_code/off2pds.m','-m')