ECE 5322
21st Century Electromagnetics

Instructor: Dr. Raymond C. Rumpf
Office: A-337
Phone: (915) 747-6958
E-Mail: rcrumpf@utep.edu

Lecture #19

Synthesizing Geometries for 21st Century Electromagnetics

Interfacing MATLAB with CAD

Lecture Outline

- STL Files
  - File format description
  - Problems and repairing
- MATLAB Topics
  - Importing and exporting STL files in MATLAB
  - Visualizing surface meshes in MATLAB
  - Generating faces and vertices using MATLAB
  - Surface mesh → 3D grid
- CAD Topics
  - Converting flat images to STL
  - Point clouds
  - Importing custom polygons into SolidWorks
  - STL to CAD file conversion
  - Exporting STL from Blender with proper units

STL File Format Description

What is an STL File?

STL – Standard Tessellation Language

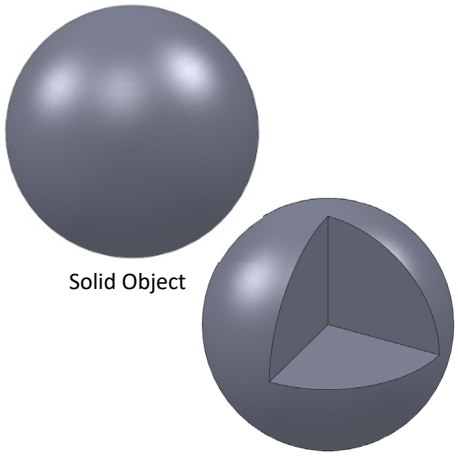
STL does not mean stereolithography file.

This file format is widely used in rapid prototyping (i.e. 3D printing, additive manufacturing). It contains only a single triangular mesh of an objects surface. Color, texture, materials, and other attributes are not represented in the standard STL file format. Hacked formats exists to accommodate this type of additional information.

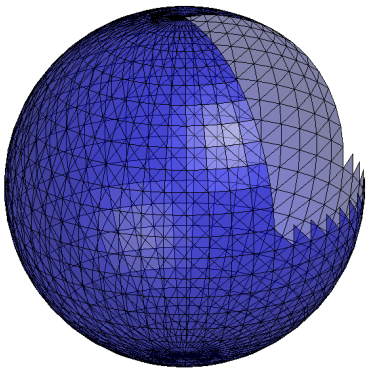
They can be text files or binary. Binary is more common because they are more compact. We will look at text files because that is more easily interfaced with MATLAB.

Surface Mesh

Despite this sphere really being a solid object, it is represented in an STL file by just its surface.



Solid Object



STL Representation

STL File Format

solid *name*

•••

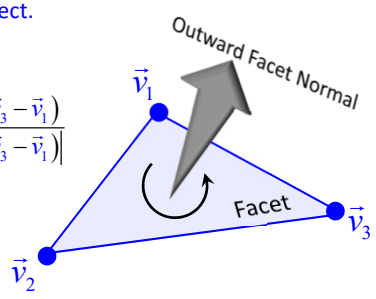
facet normal $n_x\ n_y\ n_z$
  outer loop
    vertex $v_{x,1}\ v_{y,1}\ v_{z,1}$
    vertex $v_{x,2}\ v_{y,2}\ v_{z,2}$
    vertex $v_{x,3}\ v_{y,3}\ v_{z,3}$
  endloop
endfacet

•••

endsolid *name*

This set of text is repeated for every triangle on the surface of the object.

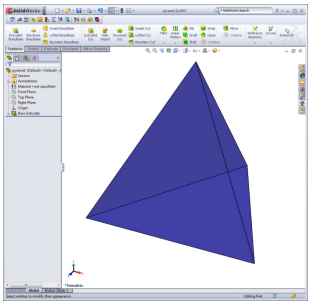
$$\hat{n} = \frac{(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)}{\left|(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)\right|}$$



Bold face indicates a keyword; these must appear in lower case. Note that there is a space in "facet normal" and in "outer loop," while there is no space in any of the keywords beginning with "end." Indentation must be with spaces; tabs are not allowed. The notation, "{…}+," means that the contents of the brace brackets can be repeated one or more times. Symbols in italics are variables which are to be replaced with user-specified values. The numerical data in the **facet normal** and **vertex** lines are single precision floats, for example, 1.23456E+789. A **facet normal** coordinate may have a leading minus sign; a **vertex** coordinate may not.

Example STL File

```
solid pyramid
   facet normal -8.281842e-001 2.923717e-001 -4.781524e-001
      outer loop
         vertex 4.323172e-018 1.632799e-017 6.495190e-001
         vertex 3.750000e-001 7.081604e-001 4.330127e-001
         vertex 3.750000e-001 0.000000e+000 0.000000e+000
      endloop
   endfacet
   facet normal 0.000000e+000 2.923717e-001 9.563048e-001
      outer loop
         vertex 7.500000e-001 0.000000e+000 6.495190e-001
         vertex 3.750000e-001 7.081604e-001 4.330127e-001
         vertex 0.000000e+000 0.000000e+000 6.495190e-001
      endloop
   endfacet
   facet normal 8.281842e-001 2.923717e-001 -4.781524e-001
      outer loop
         vertex 3.750000e-001 -1.632799e-017 0.000000e+000
         vertex 3.750000e-001 7.081604e-001 4.330127e-001
         vertex 7.500000e-001 0.000000e+000 6.495190e-001
      endloop
   endfacet
   facet normal 0.000000e+000 -1.000000e+000 0.000000e+000
      outer loop
         vertex 3.750000e-001 0.000000e+000 0.000000e+000
         vertex 7.500000e-001 0.000000e+000 6.495190e-001
         vertex 0.000000e+000 0.000000e+000 6.495190e-001
      endloop
   endfacet
endsolid pyramid
```

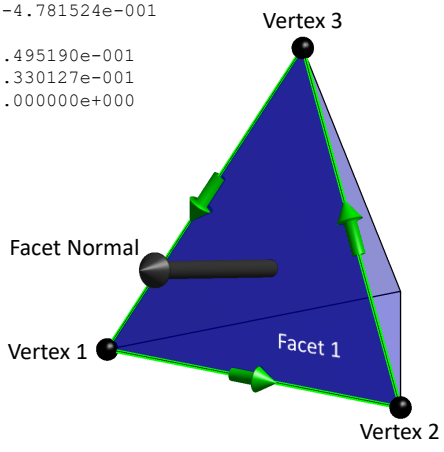


An STL file is essentially just a list of all the triangles on the surface of an object.

Each triangle is defined with a surface normal and the position of the three vertices.

A Single Triangle

```
facet normal -8.281842e-001 2.923717e-001 -4.781524e-001
   outer loop
      vertex 4.323172e-018 1.632799e-017 6.495190e-001
      vertex 3.750000e-001 7.081604e-001 4.330127e-001
      vertex 3.750000e-001 0.000000e+000 0.000000e+000
   endloop
endfacet
```



1. Facet normal must follow right-hand rule and point outward from object.
   a) Some programs set this to [0;0;0] or convey shading information.
   b) Don't depend on it*!*
2. Adjacent triangles must have two common vertices.
3. STL files appear to be setup to handle arbitrary polygons. Don't do this.

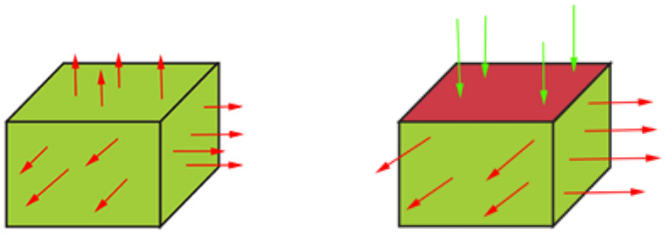
Warnings About Facet Normals

- Since the facet normal can be calculated from the vertices using the right-hand rule, sometimes the facet normal in the STL file contains other information like shading, color, etc.
- Don't depend on the right-hand rule being followed.
- Basically, don't depend on anything!

STL File Problems and Repairing

Inverted Normals

All surface normals should point outwards.

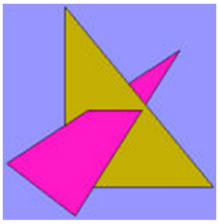


Good Bad

http://admproductdesign.com/workshop/3d-printing/definition-of-stl-errors.html

Intersecting Triangles

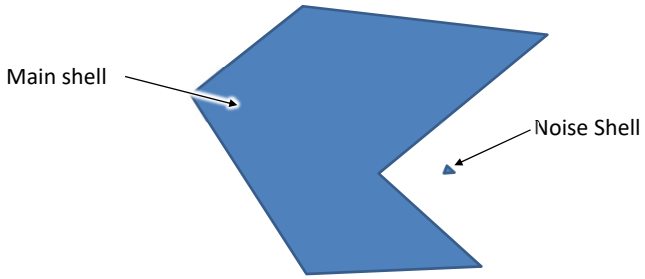
No faces should cut through each other. Intersections should be removed.



http://admproductdesign.com/workshop/3d-printing/definition-of-stl-errors.html

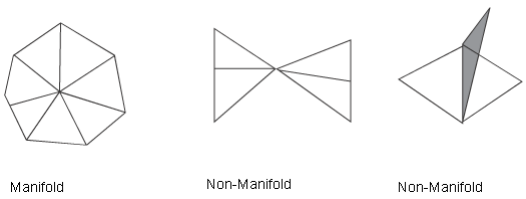
Noise Shells

A shell is a collection of triangles that form a single independent object. Some STL files may contain small shells that are just noise. These should be eliminated.



Nonmanifold Meshes

A manifold (i.e. watertight) mesh has no holes and is described by a single continuous surface.



http://http://www.autodesk.com/

Mesh Repair Software

- Commercial Software
  - Magics
  - NetFabb
  - SpaceClaim
  - Autodesk
- Open Source Alternatives
  - MeshLab
  - NetFabb Basic
  - Blender
  - Microsoft Model Repair

http://emlab.utep.edu/opensource.htm

Scroll down to "CAD Software"

Importing and Exporting STL Files in MATLAB

MATLAB Functions for STL Files

The Mathworks website has very good functions for reading and writing STL files in both ASCII and binary formats.

STL File Reader
http://www.mathworks.com/matlabcentral/fileexchange/29906-binary-stl-file-reader

STL File Writer
http://www.mathworks.com/matlabcentral/fileexchange/36770-stlwrite-write-binary-or-ascii-stl-file

How to Store the Data

We have N facets and $\leq 3N$ vertices to store in arrays.

`F(N,3)` Array of triangle facets
`V(?,3)` Array of triangle vertices

Many times, the number of vertices is $3N$. Observing that many of the triangle facets share vertices, there will be redundant vertices.

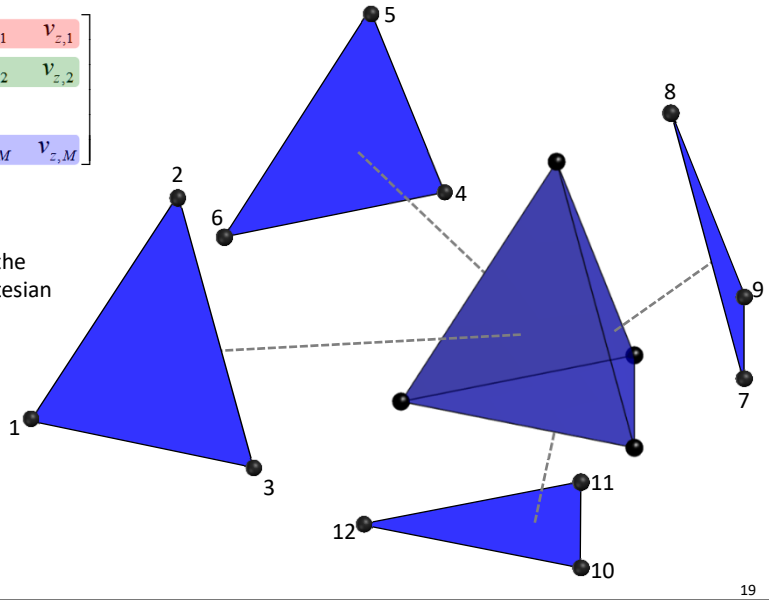
STL files can be compressed to eliminate redundant vertices, but many times they are not.

Lazy Array of Vertices (1 of 2)

$$V = \begin{bmatrix} v_{x,1} & v_{y,1} & v_{z,1} \\ v_{x,2} & v_{y,2} & v_{z,2} \\ \vdots & & \\ v_{x,M} & v_{y,M} & v_{z,M} \end{bmatrix}$$

V is an array containing the position of all the vertices in Cartesian coordinates.

M is the total number of vertices.



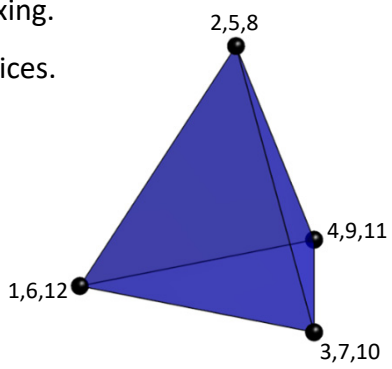
Lazy Array of Vertices (2 of 2)

There is redundancy here. Twelve vertices are stored, but the device is really only composed of four.

While an inefficient representation, this is probably not worth your time fixing.

SolidWorks exports a lazy array of vertices.

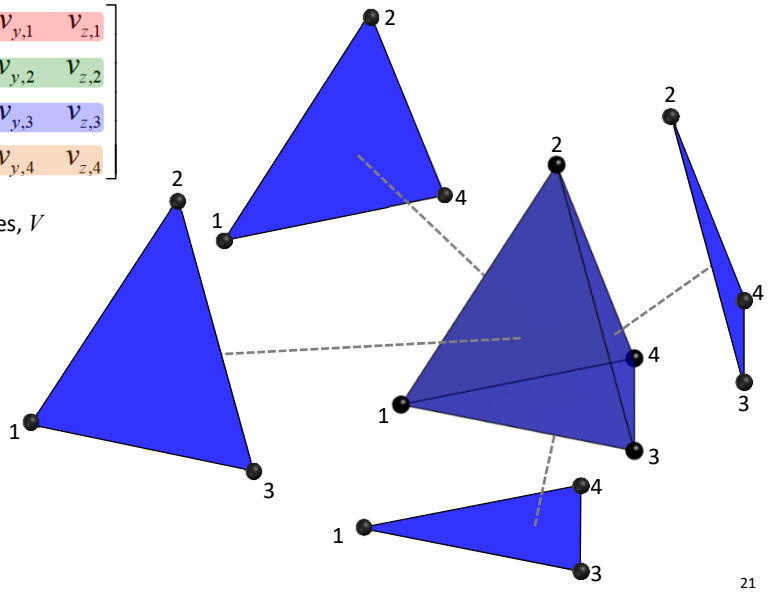
$$V = \begin{bmatrix} v_{x,1} & v_{y,1} & v_{z,1} \\ v_{x,2} & v_{y,2} & v_{z,2} \\ \vdots & & \\ v_{x,M} & v_{y,M} & v_{z,M} \end{bmatrix}$$



Compact Array of Vertices

$$V = \begin{bmatrix} v_{x,1} & v_{y,1} & v_{z,1} \\ v_{x,2} & v_{y,2} & v_{z,2} \\ v_{x,3} & v_{y,3} & v_{z,3} \\ v_{x,4} & v_{y,4} & v_{z,4} \end{bmatrix}$$

Array of Vertices, V



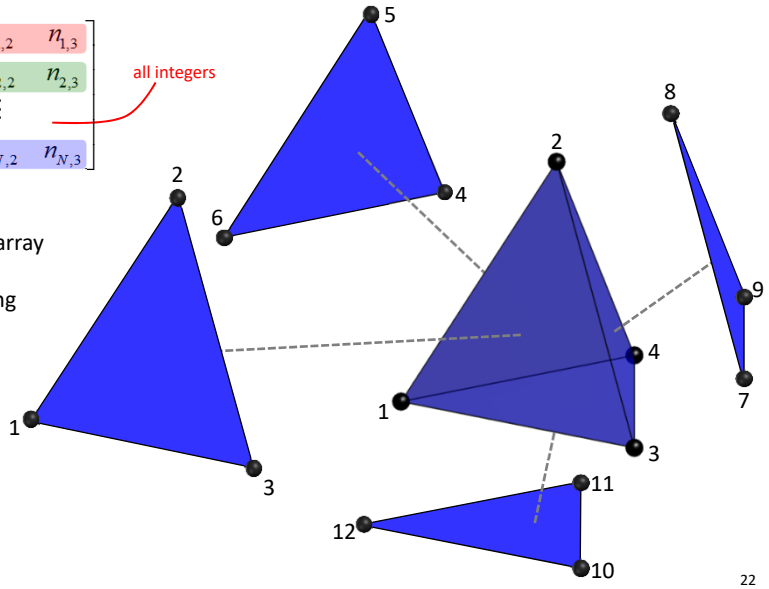
Array of Faces

$$F = \begin{bmatrix} n_{1,1} & n_{1,2} & n_{1,3} \\ n_{2,1} & n_{2,2} & n_{2,3} \\ \vdots & & \\ n_{N,1} & n_{N,2} & n_{N,3} \end{bmatrix}$$

all integers

F is an array indicating the array indices of the vertices defining the facet.

N is the total number of faces.



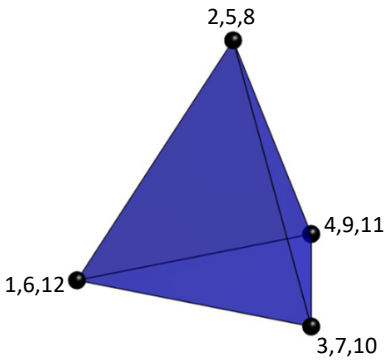
Example of Lazy Arrays

```
F =

     1     2     3
     4     5     6
     7     8     9
    10    11    12


V =

    0.0000    0.6495    0.0000
    0.3750    0.4330    0.7082
    0.3750         0         0
    0.7500    0.6495         0
    0.3750    0.4330    0.7082
         0    0.6495         0
    0.3750         0   -0.0000
    0.3750    0.4330    0.7082
    0.7500    0.6495         0
    0.3750         0         0
    0.7500    0.6495         0
         0    0.6495         0
```



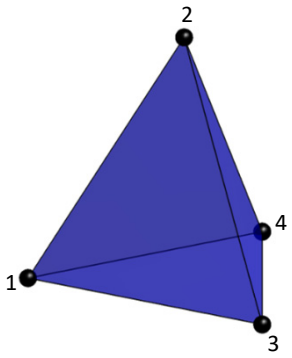
Example of Compact Arrays

```
F =

     1     2     3
     4     2     1
     3     2     4
     3     4     1


V =

    0.0000    0.6495    0.0000
    0.3750    0.4330    0.7082
    0.3750         0         0
    0.7500    0.6495         0
```



This can make a very large difference for large and complex objects.

Eliminating Redundant Vertices

Redundant vertices are easily eliminated using MATLAB's built in `unique()` command.

```
% ELIMINATE REDUNDENT VERTICES
[V,indm,indn] = unique(V,'rows');
F = indn(F);
```

Identifies unique rows in `V` and eliminates them from `V`.

Corrects indices in `F` that referenced eliminated vertices.

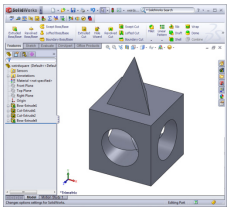
`indm` – indices of rows in `V` that correspond to unique vertices.
`indn` – Map of row indices in original `V` to new row indices in the reduced `V`.

STL Files Generated by SolidWorks

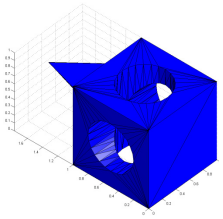
For some reason, Solidworks does not use the z-axis as the vertical axis.

For convenience, STL files can be easily reoriented.

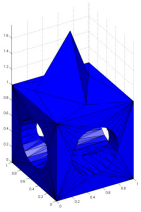
```
% REORIENT SOLIDWORKS AXES TO MATLAB AXES
Y       = V(:,2);
V(:,2) = V(:,3);
V(:,3) = Y;
```



Orientation in SolidWorks



Imported Orientation



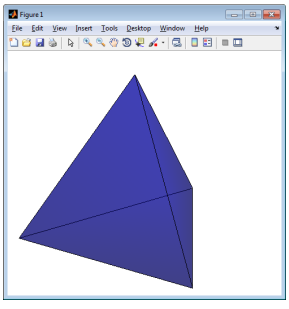
Adjusted Orientation

Visualizing Surface Meshes in MATLAB

How to Draw the Object

Given the faces and vertices, the object can be drawn in MATLAB using the `patch()` command.

```
% DRAW STRUCTURE USING PATCHES
P = patch('faces',F,'vertices',V);
set(P,'FaceColor','b','FaceAlpha',0.5);
set(P,'EdgeColor','k','LineWidth',2);
```

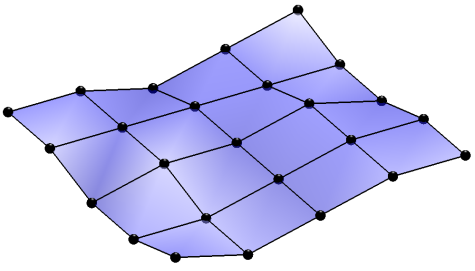


Generating Faces and Vertices Using MATLAB

MATLAB Surfaces

Surfaces composed of square facets are stored in `X`, `Y`, and `Z` arrays.

The surface shown is constructed of arrays that are all 5×5.



```
X =

   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
```

```
Y =

   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -0.6000   -0.6000   -0.6000   -0.6000   -0.6000
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
    0.2000    0.2000    0.2000    0.2000    0.2000
    0.6000    0.6000    0.6000    0.6000    0.6000
```

```
Z =

    0.1000         0    0.1000    0.2000    0.2000
         0         0    0.1000    0.2000    0.2000
         0    0.1000    0.1000    0.2000    0.1000
    0.1000    0.1000    0.1000    0.1000    0.1000
    0.1000    0.1000         0    0.1000    0.2000
```

Direct Construction of the Surface Mesh

MATLAB has a number of built-in commands for generating surfaces. Some of these are `cylinder()`, `sphere()` and `ellipsoid()`.

```
% CREATE A UNIT SPHERE
[X,Y,Z] = sphere(41);
```

Surfaces can be converted to triangular patches (facets and vertices) using the `surf2patch()` function.

```
% CONVERT TO PATCH
[F,V] = surf2patch(X,Y,Z,'triangles');
```

The faces and vertices can be directly visualized using the `patch()` function.

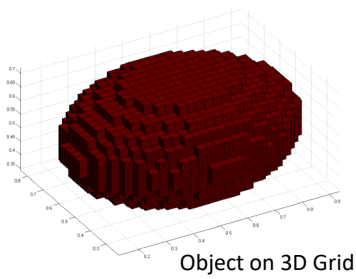
```
% VISUALIZE FACES AND VERTICES
h = patch('faces',F,'vertices',V);
set(h,'FaceColor',[0.5 0.5 0.8],'EdgeColor','k');
```


Grid → Surface Mesh

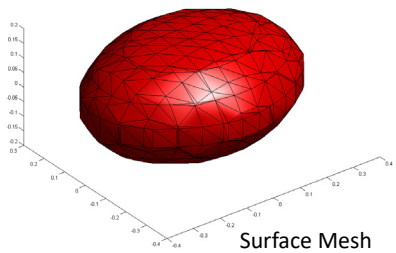
3D objects on a grid can be converted to a surface mesh using the command `isosurface()`.

```
% CREATE ELLIPTICAL OBJECT
OBJ = (X/rx).^2 + (Y/ry).^2 + (Z/rz).^2;
OBJ = (OBJ < 1);

% CREATE SURFACE MESH
[F,V] = isosurface(X,Y,Z,OBJ,0.5);
```



Object on 3D Grid

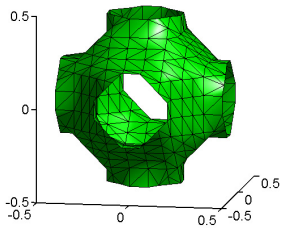


Surface Mesh

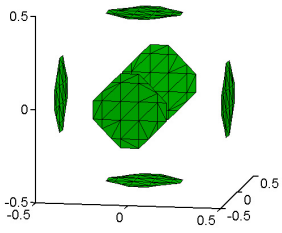
May Need `isocaps()`

When 3D objects extend to the edge of the grid, you may need to use `isocaps()` in addition to `isosurface()`.

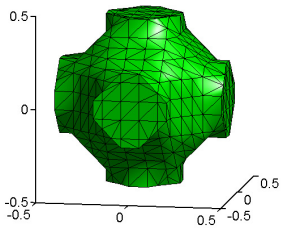
`isosurface()`



`isocaps()`



`isosurface()`
`+ isocaps()`

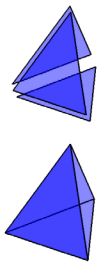


```
% CREATE SURFACE MESH
[F,V]   = isosurface(X,Y,Z,OBJ,0.5);
[F2,V2] = isocaps(X,Y,Z,OBJ,0.5);
```


Combining Faces and Vertices from Two Objects

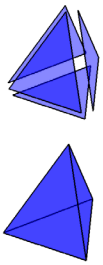
There are no functions in MATLAB to perform Boolean operations on multiple meshes. We can, however, combine the faces and vertices from two objects. Be careful this does not result in overlaps or gaps between objects.

`F1` and `V1`

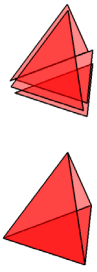


`F2` and `V2`

Correctly Combined



Incorrectly Combined

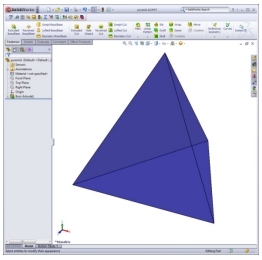


```
% COMBINE FACES AND VERTICES
F3 = [ F1 ; F2+length(V1(:,1)) ]
V3 = [ V1 ; V2 ]
```

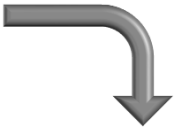
```
% WRONG WAY TO
% COMBINE FACES AND VERTICES
F3 = [ F1 ; F2 ]
V3 = [ V1 ; V2 ]
```

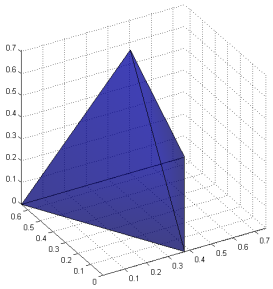
Converting Surface Meshes to Objects on a 3D Grid

Example – Pyramid



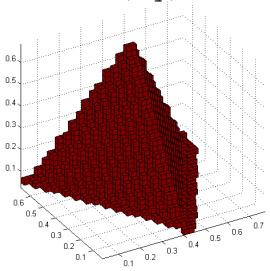
SolidWorks Model





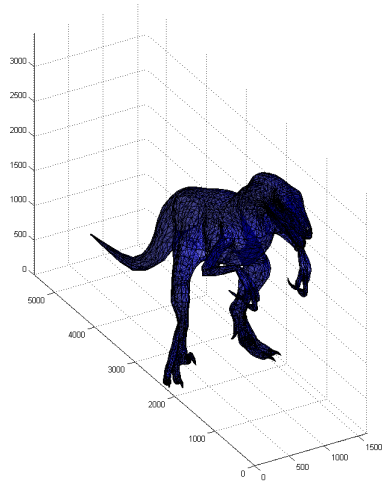
Import STL into MATLAB

ER(nx,ny,nz)



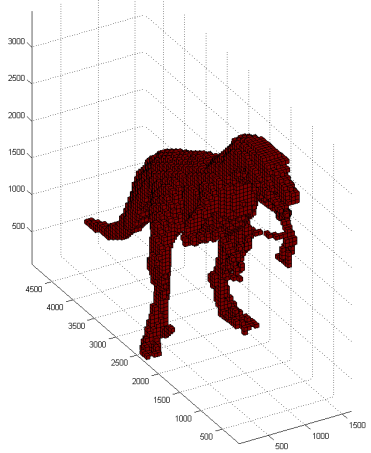
Convert to Volume Object

Example – Dinosaur



Import STL into MATLAB

ER(nx,ny,nz)



Convert to Volume Object

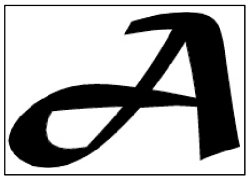
MATLAB Functions for "Voxelization" of STL Files

The Mathworks website has excellent functions for converting surface meshes to points on a 3D array.

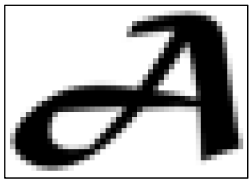
Function for Voxelization
http://www.mathworks.com/matlabcentral/fileexchange/27390-mesh-voxelisation

Converting Images and 2D Objects to STL

Load and Resize the Image



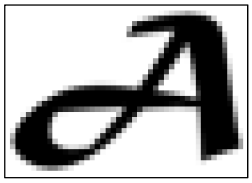
```
% LOAD IMAGE
B = imread('letter.jpg');
```



```
% RESIZE IMAGE
B = imresize(B,0.2);
[Nx,Ny,Nc] = size(B);
```

This will give us a coarser mesh in order to be faster and more memory efficient.

Flatten the Image

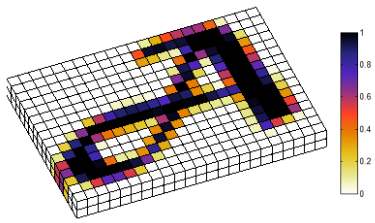


Images loaded from file usually contain RGB information making them 3D arrays. These arrays must be converted to flat 2D arrays before meshing.

```
% FLATTEN COLOR IMAGE
B = double(B);
B = B(:,:,1) + B(:,:,2) + B(:,:,3);
B = 1 - B/765;
```

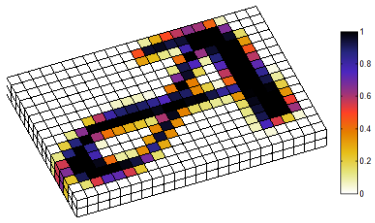

Stack the Image

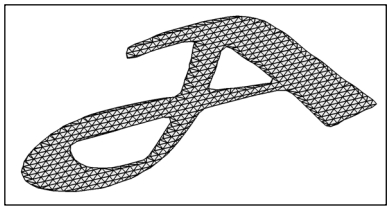
We will mesh the image using `isocaps()`, but that function requires a 3D array. So, we will stack this image to be two layers thick.



```
% STACK IMAGE
B(:,:,2) = B;
```

Mesh the Image Using `isocaps()`



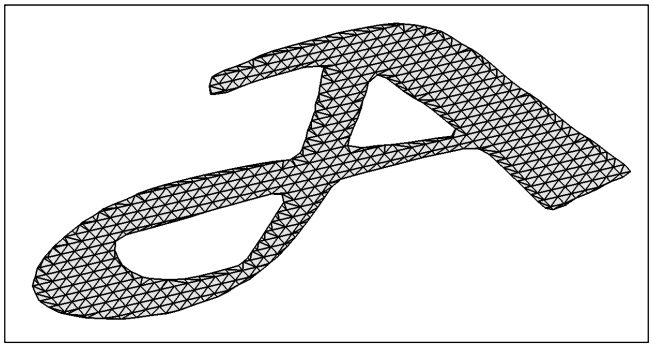


We only wish to mesh a single surface so we give `isocaps()` the additional input argument '`zmin`' to do this.

```
% CREATE 2D MESH
[F,V] = isocaps(ya,xa,[0 1],B,0.5,'zmin');
```

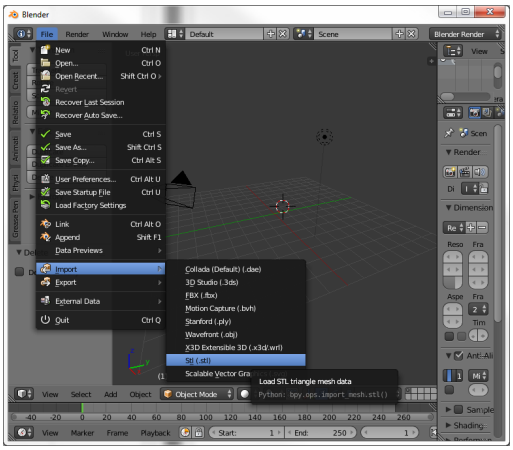

Save the Mesh as an STL File

We can save this mesh as an STL file.



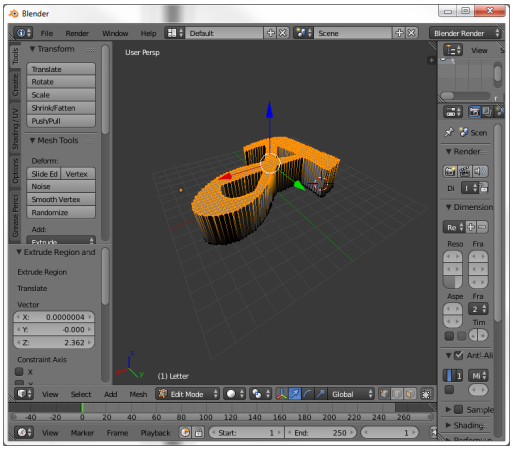
Extrude Using Blender (1 of 2)

1. Open Blender.exe.
2. File → Import → Stl (.stl)
3. Open the STL file you just created.



Extrude Using Blender (2 of 2)

1. Select the object with right mouse click.
2. Press TAB to enter Edit mode.
3. Press ‘e’ to extrude mesh.
4. Press TAB again to exit Edit mode.
5. You can now edit your object or export as a new 3D STL file.

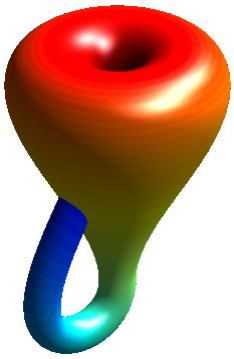


Point Clouds

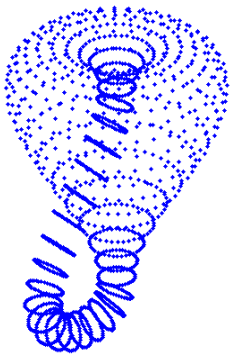
Beware of online tutorials!!!!

What is a Point Cloud?

Klein Bottle (see MATLAB demos)

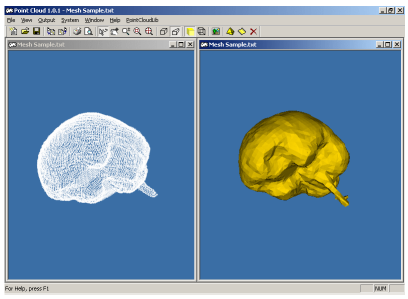


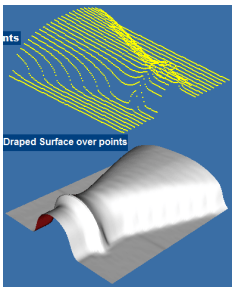
Point Cloud Description of a Klein Bottle

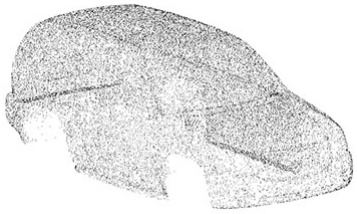


Point clouds represent the outside surface of object as a set of vertices defined by X, Y, and Z coordinates. They are typically generated by 3D scanners, but can also be used to export 3D objects from MATLAB into SolidWorks or other CAD programs.

Other Examples of Point Clouds





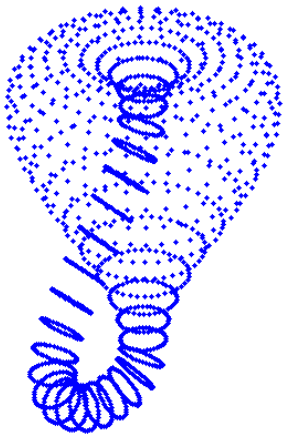


Point Cloud Data

The position of all the points in the point cloud can be stored in an array.

$$PC = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ & \vdots & \\ x_N & y_N & z_N \end{bmatrix}$$

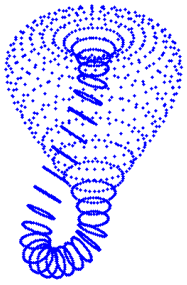
```
PC =
    0.1200    0.0000    0.7071
    0.1159   -0.0311    0.7071
    0.0311   -0.1159    0.7071
    0.0000   -0.1200    0.7071
   -0.0311   -0.1159    0.7071
   -0.0600   -0.1039    0.7071
```



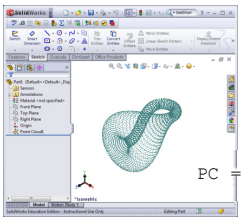
Saving Point Cloud Data to File (1 of 2)

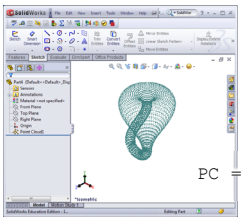
Point cloud data files are comma separated text files with the extension ".xyz"

These can be easily generated using the built-in MATLAB command `csvwrite()`.



```
% SAVE POINT CLOUD AS A COMMA SEPERATED FILE
PC = [ X Z Y ];
dlmwrite('pointcloud.xyz',PC,'delimiter',',','newline','pc');
```

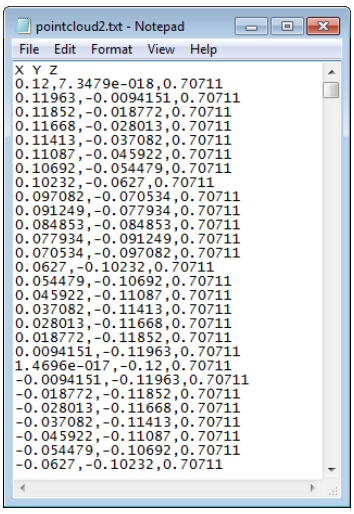




Saving Point Cloud Data to File (2 of 2)

SolidWorks wants to see an X Y Z at the start of the file.

You can add this manually using Notepad or write a more sophisticated MATLAB text file creator.



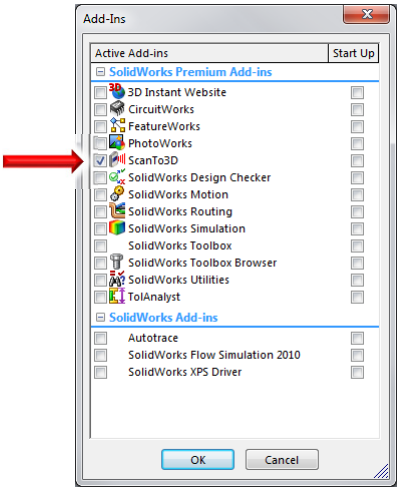
Activate ScanTo3D Add-In in SolidWorks

First, you will need to activate the ScanTo3D in SolidWorks.

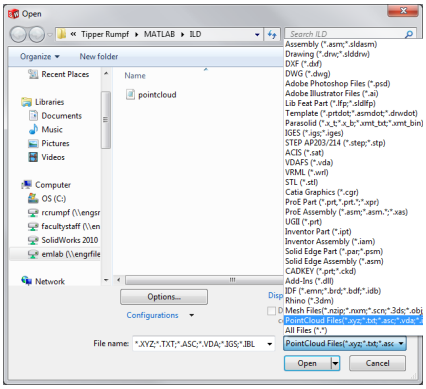
Click Tools→Add-Ins.

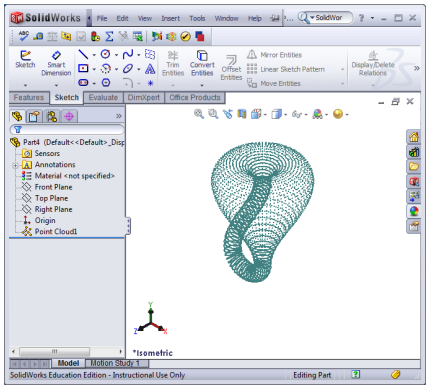
Check the Scanto3D check box.

Click OK.



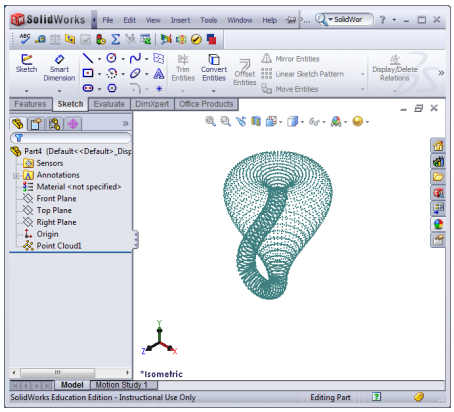
Open the Point Cloud File





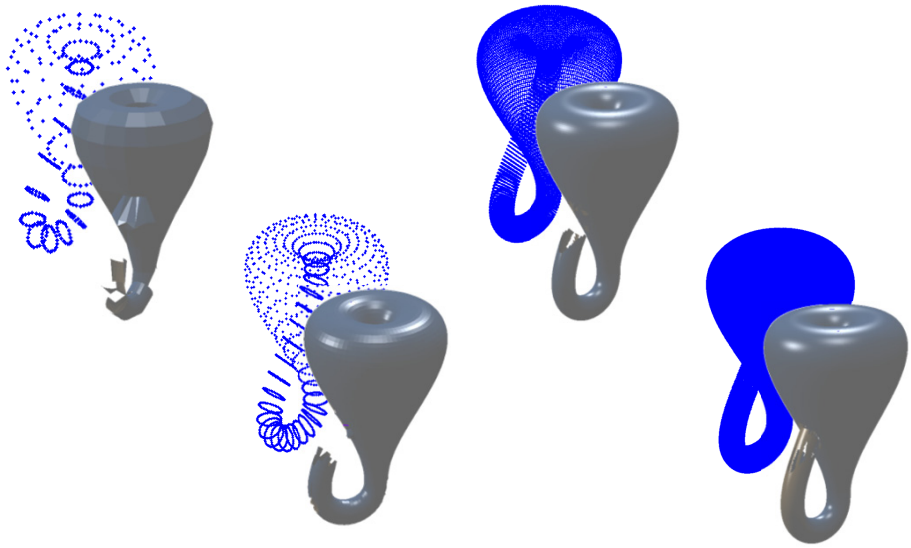
Run the Mesh Prep Wizard

Tools→ScanTo3D→Mesh Prep Wizard…



1. Run the Mesh Prep Wizard.
2. Select the point cloud.
3. Click the right array button.
4. Orientation method, select None because we did this in MATLAB.
5. Noise removal, zero assuming geometry came from MATLAB.
6. Work through all options.
7. Click the green check mark to finish.

Point Cloud Density



Final Notes on Point Clouds

- Other CAD packages have better point cloud import capabilities than SolidWorks. Rhino 3D is said to be excellent.
- Generating a solid model from the data can be done in SolidWorks. The meshed model is essentially used as a template for creating the solid model. This procedure is beyond the scope of this lecture.

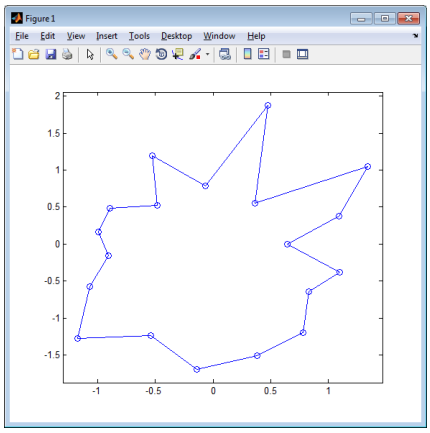
Importing Custom Polygons into SolidWorks

The Problem

Suppose we calculate the vertices of a polygon from an optimization in MATLAB.

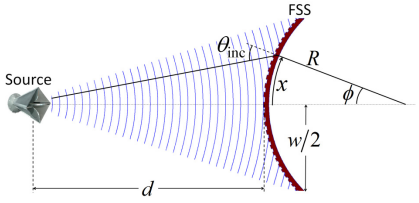
How do we important the vertices so that the polygon can be imported exactly into Solidworks so that it can be extruded, cut, modified, etc.?

There is no feature in SolidWorks to do this!!



Example Application

Placing a GMR filter onto a curved surface.



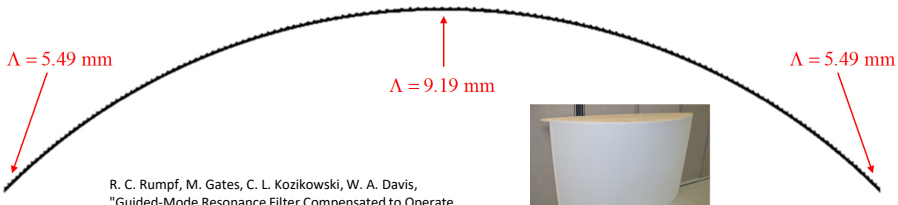
Grating period is spatially varied to compensate.

$$\theta_{inc}(x) = \frac{x}{R} + \tan^{-1}\left[\frac{\sin(x/R)}{1 + d/R - \cos(x/R)}\right]$$

$$K(x) = K_0 - k_0 n_{inc} \sin\left[\theta_{inc}(x)\right]$$

$$\Phi(x) = \int_0^x K(x')\,dx'$$

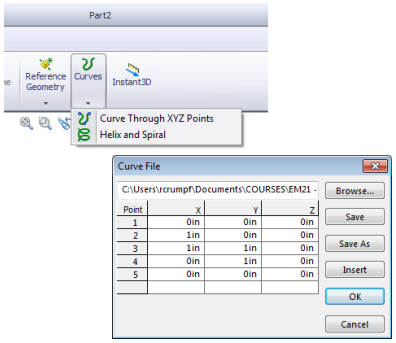
$$\varepsilon_r(x) = \begin{cases} 1 & \cos\left[\Phi(x)\right] < \cos(\pi f) \\ \tilde{\varepsilon}_r & \cos\left[\Phi(x)\right] \geq \cos(\pi f) \end{cases}$$



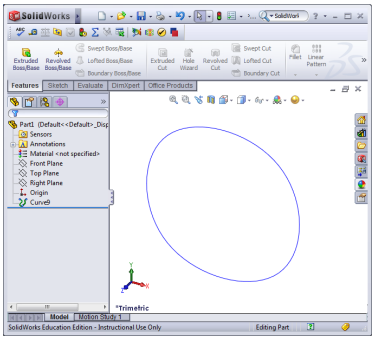
R. C. Rumpf, M. Gates, C. L. Kozikowski, W. A. Davis, "Guided-Mode Resonance Filter Compensated to Operate on a Curved Surface," PIER C, Vol. 40, pp. 93-103, 2013.

Be Careful About XYZ Curves

There is a feature in SolidWorks "Curve Through XYZ Points" that appears to import discrete points, but it fits the points to a spline. This effectively rounds any corners and may produce intersections that cannot be rendered.



This should be a square!

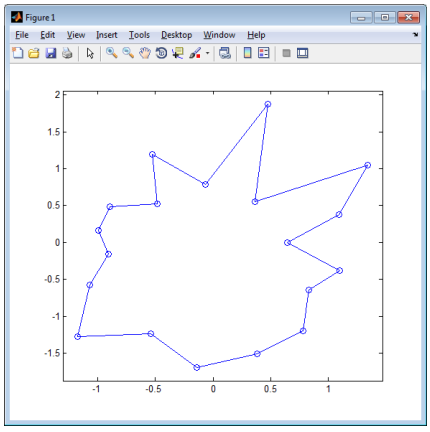


The four points are fit to a spline!

Step 1 – Define the Polygon

Create an N×3 array in memory which contains all of the points around the perimeter of the polygon. N is the number of points.

```
P =
    0.6448         0         0
    1.0941    0.3758         0
    1.3427    1.0452         0
    0.3642    0.5573         0
    0.4753    1.8765         0
   -0.0651    0.7853         0
   -0.5258    1.1984         0
   -0.4824    0.5240         0
   -0.8912    0.4822         0
   -0.9966    0.1664         0
   -0.9087   -0.1517         0
   -1.0666   -0.5774         0
   -1.1762   -1.2777         0
   -0.5403   -1.2314         0
   -0.1403   -1.6931         0
    0.3818   -1.5074         0
    0.7795   -1.1927         0
    0.8293   -0.6455         0
    1.0972   -0.3766         0
    0.6448   -0.0000         0
```



Step 2 – Save as an IGES from MATLAB

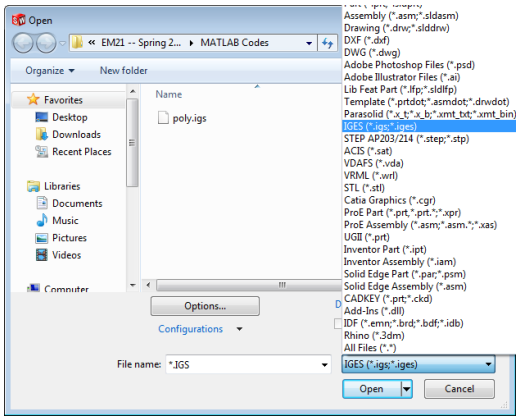
There is a function called `igesout()` available for free download from the Mathworks website. This will save your polygon to an IGES file.

```
% SAVE IGES FILE
igesout(P,'poly');
```

This creates a file called “poly.igs” in your working directory.

Step 3 – Open the IGES in SolidWorks (1 of 2)

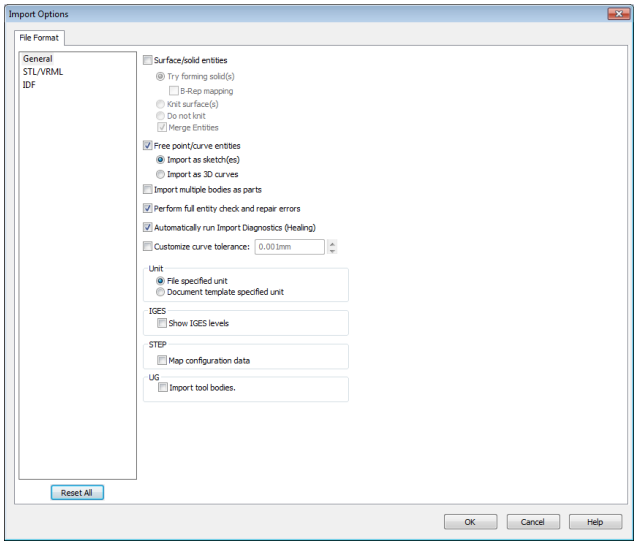
Select the IGES file type in the [Open] file dialog. Then click on the [Options…] button.



Step 3 – Open the IGES in SolidWorks (2 of 2)

1. Make sure that "Surface/solid entities" is unchecked
2. Ensure that "Free point/curve entities" is checked.
3. Click on the "Import as sketch(es) radio button.

Open the file.



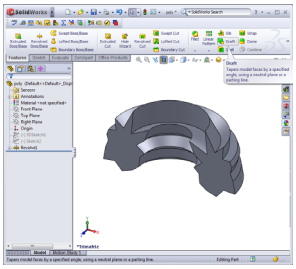
Step 4 – Convert to a Standard 2D Sketch

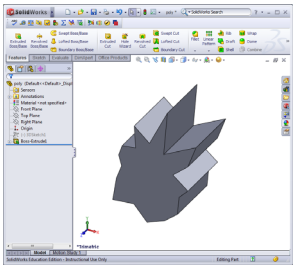
The polygon imports as a 3D sketch by default.

Edit the 3D sketch and copy.

Open a new 2D sketch and paste.

Now you can do whatever you wish with the sketch! Extrude, revolve, cut extrude, etc.

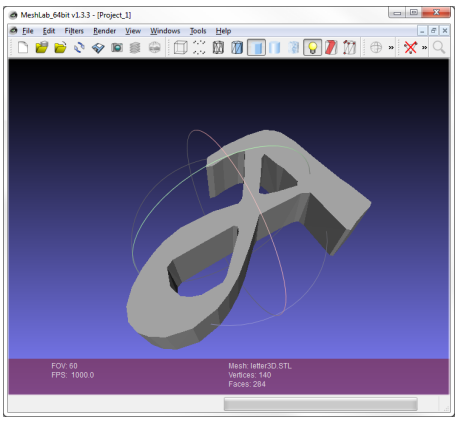




STL to CAD Conversion

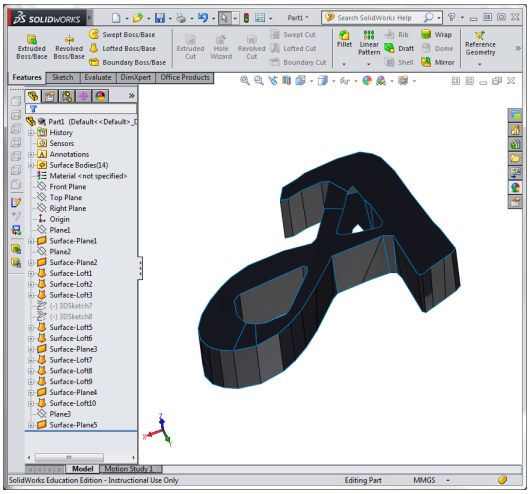
Import STL Into MeshLab

1. Open meshlab.exe.
2. File → Import Mesh
3. Open the 3D STL file you just created.



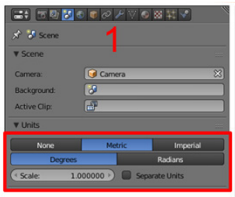
Convert to SolidWorks Part

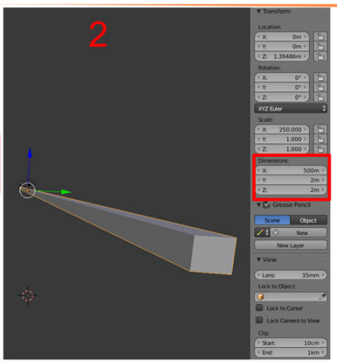
1. Open meshlab.exe.
2. File → Export Mesh As
3. Select DXF file type.
4. Save mesh.
5. Launch SolidWorks.
6. File → Open
7. Select DXF file.
8. Select 'Import to a new part as:' then '3D curves or model' then 'Next >.'
9. Select 'All Layers'

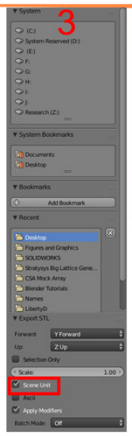


Exporting STL from Blender with Proper Units

Properly Sizing STL Files in Blender







1. Set units in Blender
2. Resize appropriately (units will be rescaled to millimeters by slicer)
3. Save and make sure scene unit is selected
4. Sizes will automatically rescale to millimeters