**HyperMesh “Fun” Sheet**

**What this guide is and is not**

This is a guide to creating an analytical mesh from an imported surface mesh, using HyperMesh, originally written by Jen Bright in 2010 and amended by Jen, Stephan Lautenschlager, and Eva Herbst since. It does not describe what finite element analysis is and is not a guide for best practice in FE models, nor is it instructions for what types of elements/loadings/analyses etc. are the most appropriate. The modelling choices that you make for your model are just that: choices for YOUR model. Depending on which solver you are using (e.g. Optistruct, Abaqus, others), you should read about the different types of boundary conditions your choices imply.

This guide will take you through the process of cleaning the surface mesh, building a 3D mesh from it, and setting it up for analysis. It’s detailed. You should probably get yourself some snacks and work through it slowly. In this case, we’re setting up a linear elastic, static model with quadratic, tetrahedral elements. There is a model on the Github page of a pig skull that has been deliberately ‘broken’ with some of the common mesh errors so that you can practice (see readme) and a PowerPoint by Andrew Cuff showing what everything will look like as you work through (although the screencaps come from an earlier version of HyperMesh, things are *mostly* in the same place still).

If you want to learn more about palaeobiological applications of FEA there are lots of cool papers out there. Some reviews to get you started are:

* [Rayfield, EJ. 2007](https://doi.org/10.1146/annurev.earth.35.031306.140104). Finite Element Analysis and Understanding the Biomechanics and Evolution of Living and Fossil Organisms. *Annual Review of Earth and Planetary Sciences* 35:1, 541-576
* [Bright, JA. 2014](https://doi.org/10.1666/13-090). A review of paleontological finite element models and their validity. Journal of Paleontology, 88(4), 760-769.

**Before you get started…**

The better the quality of your CT (I use Avizo so tend to name that software by default, but it could be from any segmentation software) work, the better quality the FEA mesh will be. It’s worth getting things as perfect as possible in Avizo before moving into Hypermesh, because once you’ve started editing the mesh, you don’t really want to have to go back and change things in Avizo and re-import. This essentially amounts to starting again with HyperMesh. HyperMesh can probably solve all your geometric crises in an emergency, but this sort of thing tends to be easier in Avizo, especially if it’s a big job.

**1. Navigation**

All the collectors (components, loads, materials, etc.) are displayed in the model tree on the left hand side of the screen, under the *Model* tab. In order to modify a collector, it must be “current”, so right-click and select “make current”. The collector will appear in bold. If you have a lot of collectors that you’re not using and the tree’s getting a little long, you can deactivate collectors under the *Entity State* tab and un-checking the “Active” box. The *Entity State* tab also contains export boxes, which become relevant later when moving the model to the solver.

Next to the model tree is a selection of buttons that control the mask function. This can also be accessed through the *tools* panel, but as it is a tool that is so frequently used, becoming familiar with these buttons speeds things up considerably. Hovering the mouse over the buttons gives a description of what they do.

Panels are displayed along the bottom of the screen. For 2D mesh editing, you will mostly need the *2D* and tools panels, the *3D* panel for tetrameshing, and the *Analysis* tab for model assembly. Beneath the panels is a command window (not always displayed) and window that displays immediate information on tasks and any error messages.

*To rotate model*: ctrl + left click

*To reset point of rotation*: ctrl + left and right click

*To translate model*: ctrl + right click

*To zoom in and out*: ctrl + mouse wheel

*To centre view on model:* ctrl + click mouse wheel

*To zoom on selected area*: ctrl + click and drag mouse wheel to select area

*To select*: shift + left click

*To de-select*: shift + right click

*To perform main panel action (indicated by whichever button in the panel is highlighted*: click mouse wheel

Once in a HyperMesh panel, things in yellow boxes are things you can select. When the box is surrounded by a blue line, it is active and anything you select will be assigned to that box for whatever action the panel is designed to perform. If you click on the yellow box, it will give you options for how to select elements by means other than clicking or selecting them in the visualisation window, e.g. by ID, by collector, etc. If there are drop-down arrows at the side of the yellow box, you can change the entity you are selecting; elements, nodes, loads, etc. Sometimes these menus have quite a lot of choice, and the menus extend on to different pages, reached by using the right and left arrows at the sides of the menus.

All of these menus disappear when you move the mouse button off them, and the mouse always centres on the menu when you open it. I mention this only because at first it can seem like the menus are jumping away from you for no reason - this is why!

Useful keyboard shortcuts:

F2: Delete panel

F5: Mask panel

F6: Edit element panel

F10: Check elements panel

Shift + F3: Edges panel

Shift + F5: Find panel

Shift + F10: Normals panel

h: Opens the online help. If you're in a particular panel, it takes you to the help pages specifically for that panel/function

f: Fits the model to the window

Esc: Returns you to the previous panel

o: Options window. This has things like the tolerance/graphics settings, etc. It's probably not a good idea to play around in here until you feel confident you know what you're doing, except for the "colours" portion. Here, you can change the background colour, which is good for making screen grabs for presentations.

m: Hides the panels. Again, this can be good for screen grabs.

**2. Importing meshes**

A variety of 3D file formats can be imported into Hypermesh. The most simple is the .stl format, a universal 3D format, which can be generated and read by most software. However, .stl files do not save shared contact surfaces and therefore require extra steps to connect components for models with more than one material (see 2B).

**2A. Importing meshes from Avizo**

For the easiest import and connection of multiple components/materials, Avizo can export models as Hypermesh Ascii .hmascii format. To load a model select *File -> import -> Solver Deck*

As *file type* select:  *HyperMeshASCII*

HyperMesh converts labels to *components*, and creates one for each contact, so for instance, to mesh the mandible you need to have mandible-exterior and mandible-teeth displayed. Meshing these contact surfaces will automatically connect the adjacent labels in your model.

**2B. Connecting meshes imported from Blender or other programs**

***NOTE THAT THIS STEP NEEDS TO BE DONE AFTER TETRAMESHING, BUT IT IS LISTED HERE BECAUSE IT IS RELEVANT TO KNOW FOR MODEL CREATION/IMPORT.***

If you do not have Avizo, you can import two adjacent components as separate .stls (for example from a program like Blender). To make the connection in Hypermesh easiest, you can make sure that both of their meshes share the same nodes at their contact surface. For example, if you are modelling a thin structure on top of another structure, duplicate the part of the bigger structure that would be the contact surface of the two, and then use that as the “base” of the thin structure. The method will also work if you want to connect two surfaces that are close to each other but do not share the same nodes – *you just need to make sure that the distance between them is smaller than the distance between two regular nodes within one of your meshes*.

Even if the node locations are the same between objects you want to connect, you need to tell hypermesh to connect the nodes. To do that, click shift + F3, select the area you want to check (shift + left mouse drag), then set the tolerance to something smaller than the distance between normal separate nodes (e.g. edge length) of your model. Select “preview equivalence” and make sure the nodes you want to merge are selected - if not, play around with the tolerance. Then once the nodes are selected, click “equivalence” and they should be merged.

**3.Tetrameshing**

**Create another component for the 3D mesh!** This way, you can go back to the 2D mesh safely if it all goes horribly wrong. This is not just necessary for backup, but also do avoid issues further along:if you tetramesh in same layer as surface model,

it doesn’t overwrite and you will end up with both tetramesh and 2D surface model meshed together in your model, throwing the error “missing material properties”. So always create a new component before tetrameshing (Collectors>Create>Component).Make sure your 3D component (which is empty for now) is active and go to *3D > tetramesh* and into the tetramesh sub-panel. Select *optimise mesh quality* and mesh speed as *gradual*, then highlight the 2D elements (hide any other layers that you don’t want to mesh) and click *mesh*. If you get any errors in the tetrameshing, it’s because your 2D mesh isn’t high enough quality, so go back and check the mesh quality (see below). The tetramesher will automatically save the elements that caused it to fail, so if you retrieve these, you’ll probably be able to see what the problem was. To retrieve them, first click “save failed”.

**Also, don't have any number characters in your component names. It confuses Abaqus.**

*NOTE: Sometimes the tetramesher will mesh the cavities in the model rather than the model itself. If this happens, delete the meshed cavity elements, go back to the 2D mesh, reverse all the normals in the mesh, and try again.*

Once you’ve got a 3D mesh, check its quality in *the tools > check elems* panel and the 3D sub-panel. The same ideas apply to 3D checks as to 2D checks. Some of them can be fixed by remeshing. Isolate elements that fail Vol. Skew >0.98, and unmask the adjacent elements. To view these failed elements, first click the little subdivided rectangle in the component hierarchy next to your mesh to turn of elements. Then click the binoculars to find the elements. In the dropdown select “elements” (nodes will be the default), then click on the yellow box that says “elements” and select “retrieve” and then “find” to view these elements. Go back to the tetramesh panel, enter the tetra re-mesh sub-panel and change the mesh speed to “interpolate”. Highlight the elements and click re-mesh. Repeat the check, and re-mesh any remaining failures as gradual.

Alternating like this can get rid of most problems, but it’s not necessarily worth it. Repeat for Tet Collapse <0.2 and aspect ratio >10. Again, the numbers are flexible and it doesn’t really matter if you don’t get them all. About 50 odd in each category has been OK in a 500,000 element model, which has led to only 30-odd distorted elements in the solver. The model runs these as warnings, but they don’t seem to have any noticeable effect.

***But if it doesn’t have a noticeable effect, why is the software complaining at me?!*** *Good question. It boils down to the difference between what is technically correct vs. what is practical, or from the solver’s perspective, what constitutes an error vs. what constitutes a warning. The equations behind the FE model expect the elements to be a certain shape, be that a perfect cube, or a perfect tetrahedron, or a perfect equilateral triangle. If you deviate from that shape, then the equations are still going to solve (probably) but the assumptions of a perfect shape will have been violated, and that introduces error. So should every element be perfectly shaped? Yes. Can every element be perfectly shaped? In a biological model, it’s not always that simple to force that assumption, unless you’re making a model via direct voxel conversion. So you should* try *to fix the really badly distorted elements, and if there are any that you don’t fix, make sure they’re not in areas of interest. If you have one bad element surrounded by 1000 good ones, it’ll probably be fine. If you have lots of bad elements all clustered together in one spot, it’s likely to create artefacts.*

**4. Mesh Quality**

*The Quick Version:*

Newer versions of HyperMesh have a button in the *3D > tetramesh* panel called “check 2D mesh”, that basically runs through all the checks below and offers some automated fixing options. (Also available in the Utility tab in the model tree [if you can’t see it, use the top bar menus *View > Browsers > HyperMesh > Utilities*], then click the “Geom/Mesh” button to take you through to another menu. Click the Quick TetraMesh button. Click on the components tab at the top of the model tree and select your mesh, then click proceed [bottom right corner]. This has the same functionality, but allows you to break down all the checks individually as per The Long Version.)  
It’s worth trying this first because it’s great if it works and can save you a lot of time (save before you do, and then be cautious and choose the ‘check’ rather than the ‘fix’ option, just to give yourself the opportunity to see what it’s about to try and do – the failed elements will be placed in sets in the model tree), but… HyperMesh often ‘fixes’ problems by just deleting the bad elements and leaving you with a hole in your mesh. The bigger and more complicated your mesh geometry, the more likely it is that you’re going to need to go in manually and fix things yourself. Hence…

*The Long Version:*

If the quality of your surfaces from Avizo is good, then you probably won’t have any of the following errors. But, if the 2D mesh imported from Avizo is poor quality it will need fixing before the 3D meshing algorithm can run. If the tetramesh function doesn’t work, it’s because there’s an error in the 2D mesh. The *2D > Quality Index* option gives a good idea of the mesh quality. If there are only a few failed elements, you can pick individual nodes and drag them to alter their position and try to improve the element shapes. This is useful to know, but not always a good idea because it alters the geometry of the specimen.

Some of it can probably be ironed out by going to the *2D > smooth*, option and selecting QI optimise from the drop-down menu for the type of smoothing. Smoothing” in HyperMesh is different to in Avizo, in that it tries to make all the elements of the same size, rather than altering the geometry. It doesn’t always work particularly well though, so might be better not to bother…

For larger areas, you can try the automesh or shrink-wrapping functions. While previous versions of this document advocated for the shrink-wrap (see much further below), the automesh function has dramatically improved and is now worth a go. Go to *2D > automesh* and select your elements. Check the QI optimise button and choose your desired element size (you can check your current element size in the *Geom > distance* panel and checking the distance between two nodes in the same element, or my going to *Tool > check elems* and then clicking on any element to display its geometric properties). Set the element shape to trias or quads and then mesh it to the current component. **It is recommended that you make a new component and make that current so that you can go back to your original mesh if this doesn’t work.** This could also take a while and might crash, so save your model first, then click it, and consider leaving it overnight.

The smoothing function may also work on the whole model. If you want to smooth the whole model, make sure the time limit box is unchecked. Again, this part can take a VERY long time for HyperMesh to work out if the mesh is big, and it often crashes. A good policy is to leave it overnight. Saving before (in case it crashes) and after this step is a good idea!

Regardless of whether you automeshed/smoothed, you’ll still want to run through the instructions below to check that everything is ok.

*Remove any duplicate elements*. Go to the *tools > check elems* panel, make sure the option on the left is set to 2D, then click duplicates. If HyperMesh finds any, click save failed in the same panel and then go to the *tools > delete*, panel. Change the menu to select elements and then select retrieve. HyperMesh will report how many elements it has selected, and this should match the number of failed elements it reported earlier. Delete them.

The next six steps are important to do IN THIS ORDER, as each stage creates more errors of the following type! Fortunately, they get easier to fix as you work down the list.

It’s worth learning all the little mesh editing techniques, but quite often, a faster approach is just to delete the offending elements and rebuild the surface (see free edges section), although if the surface is very complex this might not be a good idea! If your new surface looks messy (elements all different sizes and skewed shapes) then you can automatically clean this with the *2D > automesh* panel. Just select the messy elements, apply the appropriate element size and click mesh.

1. Intersections/Penetrations Check

This checks for 2D elements that cross over each other. The tetramesher doesn’t like this, and will report it as a “boundary error” if it encounters them. Most can be removed by performing this check. Go to *tools > penetrations*, and make sure the “include self-interference” box is ticked, then highlight all the elements and click check. This step can take a while (10-20 minutes) if the model is big.

Once the check is finished, it will open a new tab in the left hand panel telling you how many elements caused problems. Right-click on the failed threads and selected automatic intersection fix. This probably won’t fix all of them but will remove a number fairly quickly. The rest need to be fixed “by hand” by manually editing the mesh (see 2D panel instructions: detach/replace/edit elems/elem offset, etc.). Right-click and select display > review failed elements to see the intersections. If there are a lot, display > view only failed elements can be a good idea. When you come to fix them, if you click “unmask adjacent” from the toolbar down the left-hand side of the visualisation window, this can help you see the context of the intersection, and often the best way to fix it. The most likely place you will find these types of error is in regions of thin bone or thin tubes, where the element size may be larger than thickness/diameter. There’s an argument to be made here that if that’s what’s going on then you need a smaller element size to ensure you have enough nodes through the thickness to adequately capture the bending… at the very least, consider using quadratic elements, and really, you should [do a convergence test](https://doi.org/10.1002/ar.21358) to make sure.

1. Remove T-connections

T-connections are when more than two elements share an edge. They are usually not difficult to fix but there can be a lot of them, especially if you’ve equivalenced a lot of elements and weren’t careful about the tolerance settings! Go to *tools > edges* and select T-conns from the “*Find:* ” dropdown menu. Select the elements and click Find edges. Display them by going to *tools > mask > elements > by collector > ^edges* (any collector with a ^ is a temporary component that is reset every time the tool [e.g. “find edges”] is clicked), then still in the mask panel, elements > reverse, then click Mask. The will remain in the display.

**Note:** If your model has one than one material and they share a connecting 2D surface, then the boundaries between these surfaces will show as T-connections. These are safe because you’ll be building different 3D mesh components from these surfaces, so you don’t have to make any adjustments to T-connections that define these boundaries

*[An alternative, and faster, way of displaying these elements is to suppress the view of the 2D component you are editing by clicking on the little mesh symbol (a blue square) that appears next to its name in the model tree. This should leave you with the edges component only. Clicking “unmask adjacent” will then show you the same elements as the technique described above. It’s worth knowing both methods.]*

Click the “unmask adjacent” button along the left hand column to better see the nature of the T-connection. Often, they result because HyperMesh has tried to connect close regions that aren’t actually connected (i.e. it’s equivalenced the elements), in which case, simply deleting the T-connected elements will suffice. This can also happen if there’s been some overlap of your Avizo surfaces, giving the appearance of duplicate elements, except they aren’t properly connected at all nodes, which is why the earlier check didn’t find them. These may also be safely deleted. The “unmask adjacent” button is a good way to check which scenario your T-connections occur in.

Sometimes, you may want to preserve the geometry of the elements, but without the connection. To fix these, go through each of them following this procedure: Go to *2D > detach*, and make sure that element offset is set to a small number, say 0.2. This means that a gap will open between the elements when they are detached. Select all the elements on one side of the T-connected edge that you want to disconnect and click detach. If they are all selected together, they will all detach together, and you shouldn’t have to re-attach each node individually. It helps to mask each group of elements after you’ve fixed them so you can keep track of what you’ve already done.

1. Join up the Free Edges

Free edges are “holes” in the mesh. They tend to happen automatically because HyperMesh doesn’t always connect elements together when it imports them, and may also be created in mesh editing when elements are moved or deleted. They will be highlighted in the ^edges component, but you may have to zoom in quite tight to see the gaps. There is an “equivalence nodes” button in the edges panel. This will automatically fix these unconnected elements for you. It generally works pretty well, but I’ve found that it can sometimes create T-connections elsewhere in the process. I’d recommend saving before you try it!

To fix free edges manually, display them in the same way as for T-connections. Go to *2D > replace*, and make sure the “equivalence nodes” box is checked. Click on the node you want to move to highlight it, and then on the node you want to move it to, to combine the nodes. If this is going to delete elements, then HyperMesh will warn you. If you actually want an element to be deleted, may be better off using the delete function, because 1) there may in fact be more than one element there, for instance sliver elements, which may be virtually impossible to see next to bigger ones, and 2) sometimes letting the replace function delete elements makes HyperMesh crash…

When free edges arise from genuine holes in the mesh (probably from where you’ve deleted an element in fixing other errors), fill in the gaps by going to *2D > edit elems*, and toggle on create, and trias. Click the three corner nodes defining the points of the triangle you want to create.

They can also come from elements just floating freely in space. Delete these.

4. Check for Jacobian/Warpage

This seems to be where the odd quadrilateral element has snuck in. Find them by going to *tools > check elems* and click Jacobian, then save the failed elements. Find them, and go to *2D > edit elems*, and toggle on split and select divide quads and this seems to solve the problem. Repeat for Warpage elements *(Quad elements aren’t really a problem though, because the tetramesher should just split them automatically. However, it may not split them in the most sensible way, and you may also find quad elements that are ‘bent’ out of plane that you’ll definitely want to split).*

5. Fix Dihedral angles

Dihedrals occur when the angles between two elements are either too small or too large for HyperMesh to build tets between them. Check them by going to the Utility tab in the model tree (if you can’t see it, use the top bar menus *View > Browsers > HyperMesh > Utilities*), then click the “Geom/Mesh” button to take you through to another menu. Click the Quick TetraMesh button. Click on the components tab at the top of the model tree and select your mesh, then click proceed (bottom right corner). Click on the Dihedrals button in the tree. This will display both types of dihedral as elements attached to the ^feature component.

With dihedrals, honestly, the quickest way to fix them can often be to delete the problematic elements and manually rebuild the mesh in that area. However, if it’s gotten itself really twisted, this isn’t always practical because you may need to preserve the geometry of the underlying mesh, and building elements without anchor nodes from scratch in HyperMesh is surprisingly difficult to do. So when deleting the mesh is too risky, follow the instructions below.

*To fix tight angles*: Go to the *2D > elem offset* panel and check the shell offset sub-panel. Select elements on one side of the angle and set the offset to something reasonably small, like 0.5, then click on “+ offset”. Look and see if the angle gets tighter or wider. If it gets tighter, click “reject”, then click “- offset”. You have to check like this because you can’t generally tell from looking at the isolated elements which is the outside of the model, and therefore which directions positive and negative are. Whilst you’re doing this, it might be worth unmasking adjacent elements to make sure that you can’t just delete a thin sliver of bone, or alternatively that you’re not opening up an unrealistic space. Fortunately, what HyperMesh sees as an unacceptably tight angle is more severe than what solvers error out on, so don’t worry too much about them if they feel impossible to get right.

*To fix ``wide angles”*: This can be a little confusing…

When you build an element, HyperMesh assigns it a directionality, so that it can tell which are the internal and external surfaces in the model. However, it doesn’t seem to automatically notice the direction of the adjacent elements when you build an element in, meaning that sometimes elements go in “back-to-front”. HyperMesh then interprets these as two elements ~180o apart. They therefore stand out amongst the other dihedrals because they look like there’s nothing wrong with them! To fix them, you first need to know which elements are backwards. Go to the *tools > normals panel* and highlight the group of elements. Set the option to “colour display normals” and click Display Normals. Normals are displayed as red or blue (so if your component is coloured red or blue, you might want to change it to avoid the inevitable confusion), so whichever one is backwards will stand out. Blue represents the external surface and red the internal. Activate the “Orientation element” box and select one of the elements that is the right way round, then click “adjust elements”.

The only time this won’t work is if there is a “twist” in the model, in which case you may have to flip the elements manually. De-select the elements and click on the elements you want to change, then click the Reverse Normals button. Its colour should change. De-select this element, otherwise it will flip back when you move on to the next lot. It helps to de-select it, then mask it.

*NOTE: If you have more than one component, HyperMesh may read the boundary between the two components as a ^feature. Double check there are no free edges between the two, and if there aren’t, don’t worry about them.*

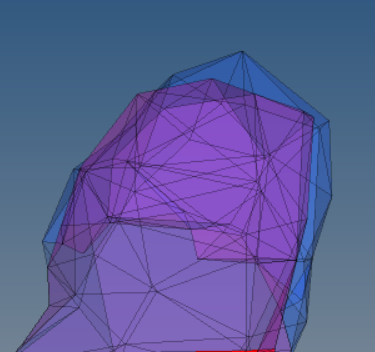
6. Element shape quality checks

If the shapes of the elements are bad, HyperMesh will struggle to produce an acceptable 3D mesh from them. Go to the *tools > check elems* panel and go in to the 2D sub-panel. Checking and finding the elements is performed in the same way as the Warpage and Jacobian checks. Aspect ratio is a good one to check first, and you should *aim* for no elems > 5 (although see note on pg. 4). There are usually some with stupidly high ARs (> 80!), which are, frankly, ridiculous and can be deleted. Others may need to be edited by splitting them or other mesh editing techniques (to split triangles, go to 2D > edit elements and in the split sub-panel, select split by line. Select the element(s) you want to split and draw the line to split them. Each click defines a change in direction for the line). Similarly, min angle should be (> 20-ish, max angle < 150-ish… the numbers aren’t set in stone).

7. Fix intersections by manually moving nodes

Sometimes, if you have multiple components, their meshes can end up intersecting each other. This can happen if your original model (for example in Avizo) had very thin structures next to each other and you then smoothed or downsampled the resulting meshes.

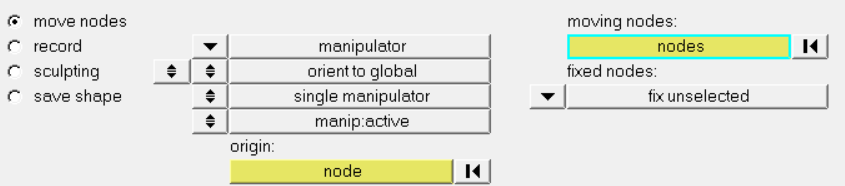
For example, here is a tooth model where the enamel and dentinelayers are intersecting each other. We want the blue layer to be surrounding the purple layer (like an onion). We can do this by manually moving nodes. Although the translate tool can do the job, the following is much quicker because it enables you to interactively move the node.



Go to Morphing (top tab menu) > Free Hand

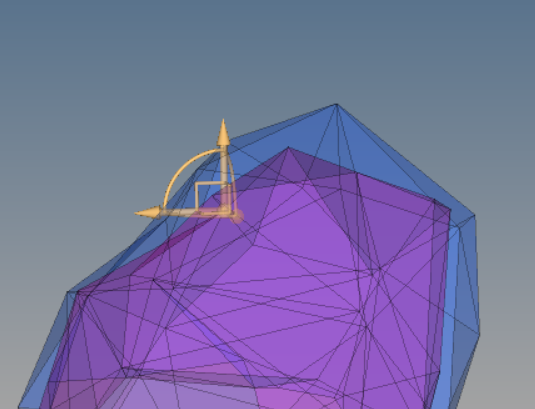
Enter the settings as follows, with fixed nodes set to “fix unselected”.

Then click on moving nodes and select your node.



(left click to select node, right click to deselect)

Use the manipulator to move the node as desired:



Once you have gone through all of these adjustments, do all the checks again to make sure that something you did hasn’t created more errors elsewhere! Dihedral and free edges are the most common culprits for this, but you’re unlikely to find too many.

It’s also worth pointing out that **FEA has no specific system of units** (see next page), so it’s worth checking the size of your model! You can measure your model by measuring the distance between two nodes using the *Geometry > distance* panel and selecting two nodes. The number reported is unit-less, but if your Avizo slices were scaled 1x1x1, it’ll probably be some multiple of the slice thickness. If you know the real size of your specimen, it’s easy to scale the model to the correct size by going to *tools > scale*, and applying a multiplication factor, either in all three directions (“uniform”) or separately in X, Y and Z.

Finishing up

Once your mesh is free of errors, it could still be substandard if there are big differences in the sizes of adjacent elements. This can happen if you have a flat area near a highly curved area, and is quite common in Avizo exports where you haven’t smoothed the mesh topology (as opposed to the geometry) before you exported the \*.stl. Alternatively, Avizo may export a mesh that is far too fine for Abaqus to solve. In these cases, you will be looking to improve the size or homogeneity of the mesh, which is great for performing convergence tests. The automesh feature is probably the best way to even this out, and it is recommended that you try this first. Alternatively, there’s shrink-wrapping, although this comes with caveats…

Shrink Wrapping

This *can be* a reallyhandy feature. The shrink-wrap feature tightly fits a homogeneously sized mesh of well-shaped quad elements over the existing 2D shell. Because the sizes are user defined, this means that you can control the mesh density, and it tends to produce a mesh with a very clean topology. However, depending on the mesh size compared to the size of the geometric features in your model, it can ‘smooth out’ details, and these details can be quite large (think of things like canals or cavities inside the bones as well as processes and crests). You should definitely check a shrink-wrapped mesh for accidentally missing geometry.

Create a new component for the shrink-wrap mesh. Make sure the shrink-wrap component is active, then go to *2D > shrink wrap*. Select the component you wish to wrap, choose an appropriate element size, then click mesh. Too coarse an element size and you’ll lose geometric, and ultimately stress/strain detail, too fine and the model will take too long to solve. Also, if you have more than one component, this may disconnect them. If this happens, you’ll have to sew up the free edges again as before.

**5. Model Assembly for Abaqus (linear elastic, static, solid)**

*Note: Oh, you want a guide for solving your models in Optistuct? Me too! It’s coming soon, once Covid is over and I can get back to the office. Watch this space.*

*UNITS: FEA has no set system of units. Use the table below to make sure everything matches.*

|  |  |  |
| --- | --- | --- |
| Length | Force | Stress/Young’s modulus |
| m | N | Pa |
| mm | N | MPa |

When creating an assembly, you need first to create a *material*. You must then create a *property* to which that material is assigned, and assign the property to the elements in your component.

1. Creating a material

Select the menu *Materials > create*.

Material name = “x\_mat”

Type = MATERIAL

Card image = ABAQUS\_MATERIAL

Click “create/edit” and scroll down to select “elastic” > “isotropic” and ELASTIC DATA CARDS = 1.

E(1) = Young’s Modulus (in MPa); Nu(1) = Poisson’s Ratio

Click “return” to accept changes.

1. Creating a property

Select the menu *Properties > create*.

Property name = “x\_prop”

Type = SOLID\_SECTION

Card image = SOLID\_SECTON

Material = x\_mat

Click “create”, then “return”.

1. Assigning properties

Select the menu *Collectors > edit > components*

Select the "assign" sub-panel, the click the “comps” button and select the component you want to assign the property to from the list. Toggle to “Property =”, double-click on it and select x\_prop from the list of solid sections. Click the “assign” button.

There is a button on the toolbar above the panels with a picture of a paintbrush over a meshed cube. You can set this to colour by properties instead of components to check that your properties were assigned to the right places.

1. Defining constraints

Select the menu *Collectors > create > loadcol* (or load collectors in some versions)

Loadcol name = “name of constraint”

Card image = INITIAL\_CONDITION

Click "create/edit”, then check the box next to “boundary”

Make the constraint loadcol current, and go to the **Analysis** > constraints panel. Select the “create” sub-panel and pick the nodes to constrain, and check which degrees of freedom you want to constrain. Click create.

1. Defining loads

Select the menu *Collectors > create > loadcol* (or load collectors in some versions)

Loadcol name = “name of force”

Card image = HISTORY

Click "create/edit”, then check the boxes next to “CLoad” and “Load case”.

Make the force loadcol current, and go to the **Analysis** > forces panel. Select the “create” sub-panel and pick nodes to apply the force to.

Specify the magnitude in N and the axis or vector you want the force to act along and click “create”.

1. Defining vectors

Go to *Analysis>vectors* and select "create". There are a number of methods available, but probably the most relevant here is *two nodes*. This creates a vector acting between two points defined by nodes, e.g. muscle insertion points on the cranium and mandible. If you're trying to project a vector to a space where no nodes currently exist, then go to the Geom panel, and use the node panels (found on the left) to create nodes that you can use, either by typing in coordinates (nodes>type in), or placing a node between two or three others (distance>two nodes/three nodes, then click "nodes between"). Once you've created these nodes, you can move them into position using the tools>translate panel.

1. Defining steps

Go to the *Analysis > load steps* panel.

Name = “name of step”

Click on “Loadcols =” and select all the forces and constraints you want to include in the analysis.

Click “create”, then right-click on the loadstep in the model tree and go to "card edit". Select “static” from the dropdown menu that appears. Click return.

If your model has multiple loading steps, simply create many load steps. You can combine/remove them in Abaqus after the analysis has run.

**6. Exporting to Abaqus**

To export to Abaqus, the file needs to be saved as an \*.inp file. Go to *File > export > model*, and select *Abaqus* and *standard3D* as the export types. You need to specify the file path name in full, ending with .inp. Set the select entities to export dropdown to “custom”, and choose which entities to export from the Entity State tab. Click export… done.

To run the job, you can either import the \*inp into Abaqus, or run it from the cmd window. I prefer the latter, because it’s faster and takes less computer power.

To do this, go to the Windows start menu and choose “Run” from the list, then type cmd into the box and hit enter.

When it opens, first, you need to specify the folder that the \*.inp file is in. Type:

**cd C:\*folderpathway***

but don’t mention the actual \*.inp file yet.

Then type:

**abq6141 interactive job=filename**

and press enter. “abq6141” is the version of Abaqus (e.g. 6.141); if you’re running a different version, change the numbers appropriately. Don’t include the suffix .inp in the filename. This line is white-space sensitive, but not case sensitive.

The progress of the file is reported in the command window… to an extent.

If it’s taking a long time to solve, look in the \*.msg file that Abaqus writes (when you run from the window, the solver files are written to the folder you’ve already specified, rather than the default Abaqus temp folder). The \*.msg file will report the progress of the analysis, so you can look here to see if there are any constraint problems preventing the model from solving. Read it to find out, because Abaqus will try in vain to resolve these constraint issues for a long time before realising it is futile! Once it’s worked out the errors, they’re written in the \*.dat file. You can use this to identify which nodes or elements are failing by ID number, and then isolate the failed entities from the mask panel in HyperMesh to fix them.

**Queuing jobs**

If you’ve been really hardcore and made lots of models, but don’t want to sit around running each one manually, then you can queue the jobs to start automatically.

Open a text editor (e.g. notepad) and write:

**call abq6141 interactive job=filename1**

**call abq6141 interactive job=filename2**

**call abq6141 interactive job=filename3**

etc…. (syntax as above).

Then save the file as “runJobs.bat” and put it in the same folder as the \*.inp, and double-click it to run the \*.bat as an executable.

*NOTE: This can backfire, because if one job stops due to errors, it’ll stop the whole queue, and you can end up not solving models further down the list.*

**7. Miscellaneous**

Taking a high-quality screen capture

Go to the *Preferences > postscript menu*

Check the box next to colour, and set options to “enhanced”, and move the slider on “jpeg quality” up to 100%. The default picture sizes are for A4 sized images. If it’s not already, open the command window by going to the *View > command window*.

In the command window, type \*jpegfile to write a jpeg screen grab to My Documents. The jpeg will be automatically assigned a file name and number in the format “imageX.jpg”

***“But jpegs are lossy, I don’t want a jpeg!”***

*Preach. Alas, this is what you’re getting. I don’t make the rules.*

Writing Macros

These are handy if you find that there’s a particularly repetitive or precise task that you need to perform. For instance, my specimen was not correctly aligned with the scanner gantry. Rather than try to create a local coordinate system, which can be horrendously difficult in HyperMesh, I rotated the model until it lined up, then saved a macro that would perform this exact rotation in other models.

Firstly, you need to delete the command.cmf file from My Documents. Open HyperMesh and do everything that you want to have included in your macro. Then, look in My Documents again; command.cmf will have been rebuilt, and contains a list of everything that you have performed in HyperMesh, with the most recent actions at the bottom. Delete everything about opening the model file from the beginning, leaving only the instructions that you want to write into the macro. Then, from the remaining text, delete all parentheses and replace all the commas with white space. Also, you need to insert white space after every word in the file (this will seem less intimidating when you see it!). Save the file as a \*.tcl. This is your macro. And yes, it’s pronounced ‘tickle’. You just tickled HyperMesh. Close HyperMesh. When you want to run your macro, right-click in the command window at the very bottom of the screen and select load file, then open your macro to run it.