

Full Throttle STEM™ Safety

Challenge Problem and Resources



Developed by:

The teachers, students, and mentors in the
Gaming Research Integration for Learning Laboratory™ (GRILL™)
Summer 2014

TABLE OF CONTENTS

1.	CHALLENGE PROBLEM: FULL THROTTLE STEM™ SAFETY.....	3
1.1.	THE TOOLS.....	3
1.2.	THE CHALLENGE	3
2.	TUTORIALS OVERVIEW.....	4
3.	BIOHARNESS 3 SETUP, USE, AND DATA ANALYSIS	4
3.1.	SETUP	5
3.2.	USING THE BIOHARNESS	5
3.3.	DATA RETRIEVAL	6
3.4.	DATA ANALYSIS.....	6
3.5.	RESOURCES	9
4.	DUAL-BOOTING UBUNTU WITH WINDOWS ON A SURFACE PRO 2 TABLET.....	9
4.1.	CREATING A BOOTABLE VERSION OF UBUNTU.....	9
4.2.	PARTITIONING THE C DRIVE FOR UBUNTU	9
4.3.	DISABLING WINDOWS 8 SECURE BOOT CONTROL.....	10
4.4.	INSTALLING UBUNTU	10
4.5.	BOOTING INTO UBUNTU	11
4.5.1.	If a Blue Screen Appears on Startup	12
4.5.2.	If the Device Boots Directly Into Windows	12
4.6.	FIXING UBUNTU’S WI-FI PROBLEMS	12
5.	FINITE ELEMENT ANALYSIS USING ADINA SOFTWARE	13
5.1.	FINITE ELEMENT ANALYSIS OVERVIEW	13
5.2.	UNDERSTANDING THE PHYSICAL PROBLEM	14
5.3.	CREATING A FINITE ELEMENT MODEL OF THE PHYSICAL PROBLEM	18
5.3.1.	Defining model control data	18
5.3.2.	Defining model geometry	18
5.3.3.	applying loads and boundary conditions.....	19
5.3.4.	Managing material definitions.....	20
5.3.5.	Meshing.....	21
5.4.	SOLVING THE MODEL / POST-PROCESSING	22
5.5.	INTERPRETING THE RESULTS AND REFINING THE MODEL.....	25
6.	LOADING UBUNTU ONTO A FLASH DRIVE AND USING PUPIL LABS SOFTWARE	25
6.1.	SETTING UP UBUNTU	25
6.2.	INSTALLING PUPIL CAPTURE.....	26
6.3.	INSTALLING PUPIL PLAYER	26
6.4.	USING PUPIL CAPTURE AND PUPIL PLAYER	27
6.5.	PATCH TO RUN PUPIL CAPTURE OFF OF A USB HUB.....	28

1. CHALLENGE PROBLEM: FULL THROTTLE STEM™ SAFETY

In indoor go-karting facilities, safety is a top priority. Many are licensed amusement operators and adhere to state guidelines. In an effort to monitor and increase the safety of customers, karting companies intend to evaluate their current safety implementations, including track and barrier materials, visual cues, and physiological impact.

This challenge is a service learning opportunity in which students will evaluate and identify ways to improve customer and personnel safety in an indoor go-karting facility. This evaluation will examine physiological impact, eye tracking, go-kart parts, and track materials and barriers.

1.1. THE TOOLS

The tools needed include a BioHarness, data processing software, such as MATLAB, eye tracking hardware, 3D modeling software, such as SolidWorks, and a simulation crash software like ADINA.

1.2. THE CHALLENGE

Provide a meaningful evaluation utilizing Modeling and Simulation technologies that will provide solutions to enhance customer safety in go karting facilities.

2. TUTORIALS OVERVIEW

Wright Scholars, in collaboration with educators and the GRILL™ team, created the tutorials described below as possible solutions to solve the challenge problem. At the time of creation these were working tutorials; however, with software updates and changes in technology, additional steps may be required. Teachers are encouraged to communicate any issues, problems, or suggested changes to these tutorials to ensure the dissemination of helpful materials to support challenge problem implementation. Below is a description of each tutorial, followed by the full tutorials.

Table 1: Tutorials Provided for Challenge Problem

Challenge Problem	Tutorials
Full Throttle STEM™ Safety	Bioharness 3 Setup Use and Data Analysis (Section 3), Dual Booting Ubuntu with Windows on a Surface Pro 2 Tablet (Section 4), Finite Element Analysis Using ADINA Software (Section 5), Loading Ubuntu and Using Pupil Labs Software (Section 6)

3. BIOHARNESS 3 SETUP, USE, AND DATA ANALYSIS

The BioHarness is a wearable device that measures a person's skin temperature, ECG, breathing, and posture. It can wirelessly transmit the data to a computer in real time via Bluetooth or data can be internally recorded and later transferred to a computer. This tutorial will address the internal logging method and analysis of the ECG data. The Bio Harness website <http://zephyranywhere.com/zephyr-labs/development-tools/> offers various software tools that can be used with the BioHarness that are not discussed here.

Suggested materials for this tutorial include a BioHarness garment, the BioHarness module, the cradle and its USB cord, and a computer.

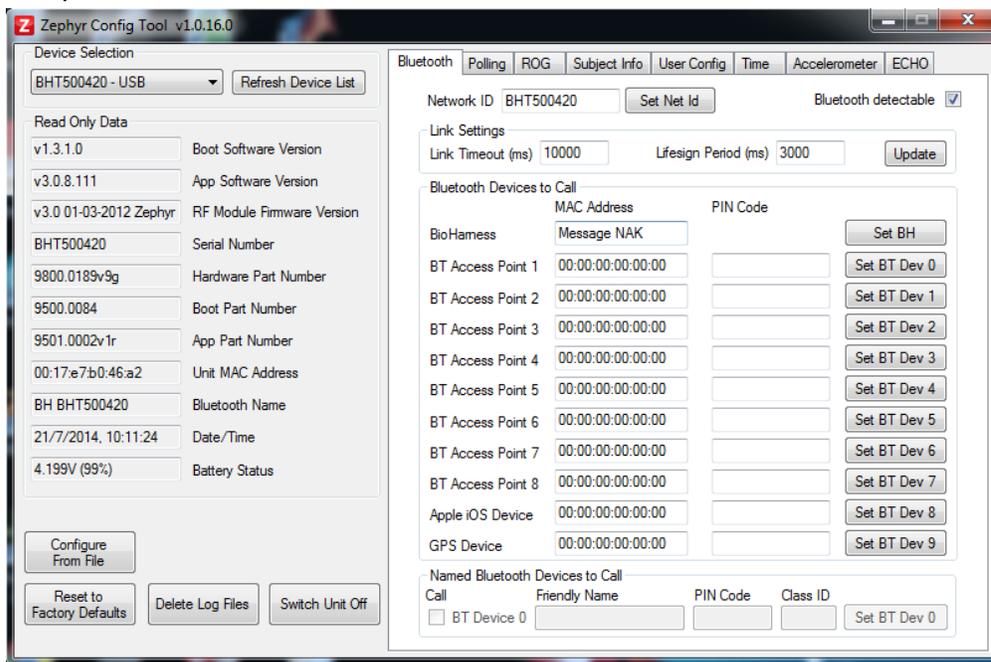
This tutorial includes the following topics:

- Setup
- Using the BioHarness
- Data Retrieval

- Data Analysis
- Resources

3.1. SETUP

1. Download and install the latest version of the BioHarness Log Downloader and the Zephyr Config Tool from the following website:
<http://zephyranywhere.com/zephyr-labs/development-tools/>
2. Download and install the USB device driver from the following website: www.zephyranywhere.com/support/downloads.
Note: Be sure to download the version that is for BioHarness 3.
3. Plug the cradle into the computer and insert the puck into the cradle.
4. Go to device selection in the Config Tool window and select the string of letters and numbers (not COM1).
5. Set the time under the time tab by pushing the Set date/time button to sync with the computer's clock.



6. If acceleration and posture data is relevant, go to the accelerometer tab and select a garment type under Presets. This tool also allows other adjustments, like sampling rate, to be made as well.

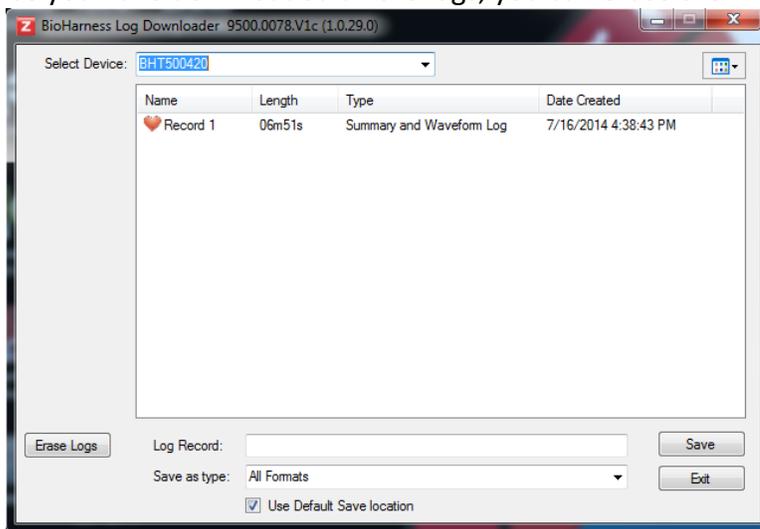
3.2. USING THE BIOHARNESS

1. Make sure the puck is charged. It can be charged from a computer or USB outlet using the cradle. The puck will have a 90% charge when the red charging light stops blinking.
2. Snap the puck into the BioHarness garment after selecting the garment type with the Config Tool as mentioned above. Push and hold the center power button until the lights light up. The device is now recording.

3. Once finished, take off the garment and turn off the puck by holding the power button until all of the lights light up.

3.3. DATA RETRIEVAL

1. Plug the cradle into the computer and insert the puck into the cradle.
2. Open the Log Downloader with the desktop icon you made earlier and click the drop down Select Device menu. You should see a device name made up of letters and numbers. Select it and a list of downloadable logs should display (if you have recorded any).
3. Select the logs you want, pick a format (.CSV is the Excel spreadsheet format) and click the save button to download the logs. If you do not want to use the default save location, uncheck the box and you will be prompted to select a location.
4. Once you have downloaded all the logs, you can erase them from the puck.



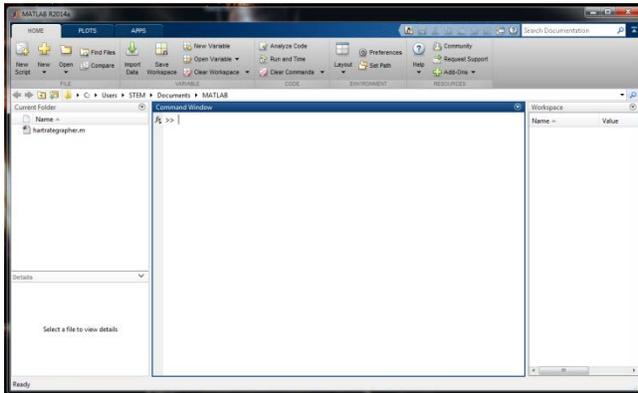
3.4. DATA ANALYSIS

Logged BioHarness data can be analyzed in various ways, but this tutorial will cover the MATLAB method of finding the heart rate from ECG data. The supplied code in its present form will not work in FreeMat, although it should be possible to rewrite this code for FreeMat. If you do not already have MATLAB, which requires a paid license, you will need to download and install it from the following website:

https://www.mathworks.com/programs/nrd/matlab-trial-request.html?ref=ggl&s_eid=ppc_4341

1. Open MATLAB and click the New Script button.

2. Copy the following code and paste it into the script that you just opened. Be sure not to copy any text that is not part of the code.



```
clc %Clearing the command window
clear all %Clearing all variables
close all %Closing all open figure windows

ecgRaw=dlmread('ECG.txt'); %Loading in ECG data and storing it
sampfreq=250; %Storing the sampling frequency in Hz
ecg=ecgRaw'; %Transposing the ECG vector

endtime = (length(ecg)-1)/sampfreq; %Calculating the length in seconds of the ECG sample
t=[0:1/sampfreq:endtime]; %Creating a time vector

%Plotting ECG vs. Time (first minute)
figure(1)
plot(t,ecg)
title('Plot of ECG')
xlabel('Time (s)')
ylabel('Voltage (mlliVolts)')

%User input for start time to throw out bad data
Tstart = input('What time (in seconds) would you like the sample to start? ');

%Plotting ECG vs. Time (last minute)
figure(2)
plot(t,ecg)
title('Plot of ECG')
xlabel('Time (s)')
ylabel('Voltage (milliVolts)')

%User input for ending time to throw out bad data
Tend = input('What time (in seconds) would you like the sample to end? '); %length(t)

%Cutting down ECG based on user input
startindex = find(t==Tstart); %determining the index of the start time
endindex = find(t==Tend); %determining the index of the end time
newecg = ecg(startindex:endindex); %Cutting down ecg data
```

```

newtime = t(startindex:endindex); %Cutting down time vector

%Plotting cut down ECG data
figure(3)
plot(newtime,newecg)
title('Plot of ECG')
xlabel('Time (s)')
ylabel('Voltage (milliVolts)')
axis([Tstart Tend 1900 2600])

%User input for minimum magnitude to be considered a QRS peak
minheight=input('What is the minimum peak height (in milliVolts) you would like to consider
    for an R wave? ');
locs = []; %Creating empty vector for location indices of peaks
j=1; %creating a counting variable for locs vector

%for loop to sort through each index of newecg to find peaks
for i = 2:(length(newecg)-2)
    if newecg(i)>newecg(i-1) && newecg(i)>newecg(i+1) %if the value at that index is greater
        than the point before and after it
            if newecg(i)>minheight %if the value at that index is also greater than the user
                entered minimum magnitude
                    locs(j) = i; %store that index value in the locations vector at index j
                    j=j+1; %advance counter variable
                end
            end
        end
    end

tpeaks = t(locs); %determining time where peaks occurred
d=diff(tpeaks); %finding the time difference between peaks
meandiff=mean(d); %finding the mean time difference (seconds per beat)
meanheartrate=(1/meandiff)*60; %Converting seconds per beat to beat per minute

fprintf(1,'The average heart rate of the subject is %.2f beats per minute (bpm)\n',
    meanheartrate);

```

3. Save the file as an .m file (the default) and minimize MATLAB.
4. Open the .csv file containing the BioHarness data in Excel. You will need to copy the column with the ECG values and paste it into Notepad. To easily highlight an entire column in Excel, click the letter at the top. Do not copy the timestamp column.
5. Once in notepad, remove the ECG label from the top of the column. Save the file as ECG.txt. This name and file extension must be used for the file to work with the script.
6. Go to your MATLAB window and find the folder selection bar. The folder that contains your .txt data file must be selected. If necessary, click the folder icon with the green arrow and browse for the correct location.
7. Go to the .m file that you made and click the play button one time.
8. Follow the prompts that appear in the command window. To determine when the sample should start and end, look at the graph of your data set supplied by MATLAB and decide where the plausible dataset appears to begin and end. Often, the beginning and end of the measurement will contain irregular and invalid data, and this should be excised from the dataset before the heart rate calculation is made. Setting the minimum peak height tells the

program how much the voltage peaks should have to rise to count as a heartbeat. This helps prevent noise from increasing the measured heart rate. A good setting for this would be 2200-2350.

9. The program will now output the calculated average heart rate!

3.5. RESOURCES

The complete user's manual can be found at the following

URL: http://zephyranywhere.com/media/pdf/BH_MAN_P-BioHarness3-User-Manual-FCC_20120912_V01.pdf.

4. DUAL-BOOTING UBUNTU WITH WINDOWS ON A SURFACE PRO 2 TABLET

This tutorial documents how to install Ubuntu on a Surface Pro 2 Tablet alongside Windows 8.1. It will guide you through prepping the tablet for install, installing Ubuntu, and specifying boot order afterwards.

The materials required include a Microsoft Surface Pro 2 Tablet, a wired USB hub, a flash drive of approximately 8 GB, a USB keyboard, and a USB mouse.

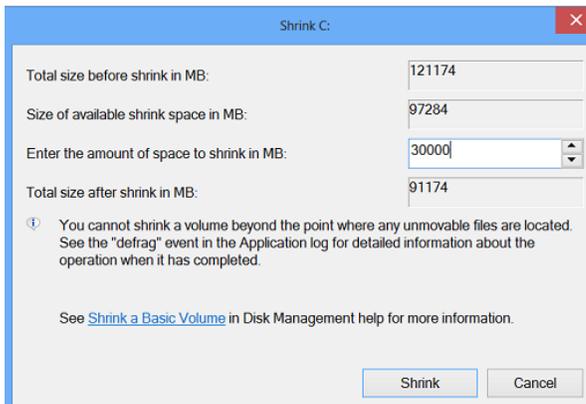
4.1. CREATING A BOOTABLE VERSION OF UBUNTU

1. On the tablet, download 64-bit Ubuntu desktop from the following website: <http://www.ubuntu.com/download>
2. Next, download Universal USB Installer from this website: <http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/>
3. Plug the flash drive into the tablet and follow the instructions on that page to install Ubuntu on the flash drive.

4.2. PARTITIONING THE C DRIVE FOR UBUNTU

Make sure you have at least 20 GB of space free on the C Drive before doing this section of the tutorial.

1. Open the charm by swiping to the left from the right side of the screen. Tap on Search. Search for Control Panel and open it.
2. In the Control Panel, select View by: Small Icons.
3. Select Administrative Tools -> Computer Management.
4. From the menu on the left side, select Disk Management.
5. Right-click on the C Drive. To right-click with the pen that comes included with the tablet, press the button on the side of the pen. A window like the following should appear.



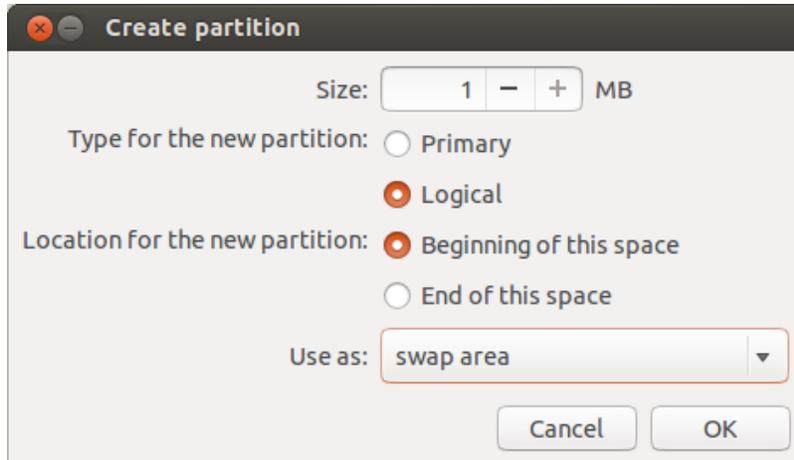
6. Select Shrink Volume and type in 20,000 MB in the field that is editable and click Shrink. You should now have approximately 20 GB of unallocated space in the C Drive. Exit out of Computer Management and Control Panel.

4.3. DISABLING WINDOWS 8 SECURE BOOT CONTROL

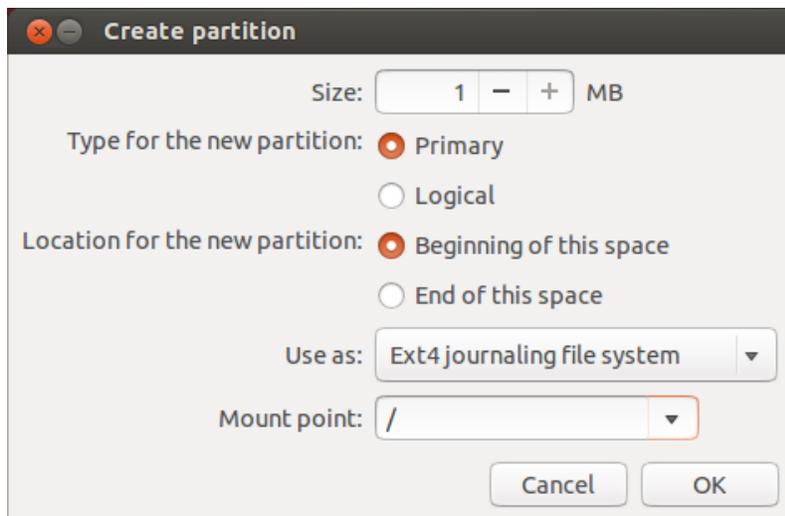
1. Open the charm and go along the following path: Settings > Change PC Settings > Update and Recovery > Recovery. Underneath Advanced startup, restart the tablet. A light blue screen should display.
2. Open Troubleshoot > Advanced options > UEFI Firmware Settings. Restart when prompted. A black screen should display.
3. Disable Secure Boot Control, exit, and restart the tablet.

4.4. INSTALLING UBUNTU

1. Plug in the USB hub, and connect the flash drive, keyboard, and mouse to the hub.
2. Go to advanced startup and restart the tablet. Select Use a device. Select your flash drive. When the option appears, use the keyboard to select to install Ubuntu.
3. Go through the installation, without installing any extra third party software, until you get to the point where you are asked where to install Ubuntu. Select something else.
4. Locate the free space you created earlier. You should be able to recognize it by its size.
5. Select the free space and click the + button underneath the drives.
6. Set the size to 2048 MB, make the partition logical, and locate it at the beginning of the space. Designate it as swap area. You should have a screen like the image below. Click OK.



7. Select the remaining empty space, which should now be somewhere around 17.5 GB. Set the size to the maximum. Set the partition as a primary type, and use it as Ext4 journaling file system. Lastly, choose / as the mount point. It should look like the below image.



8. Once you create the partition, select it from the list of partitions, and install Ubuntu. Restart the tablet when prompted.

4.5. BOOTING INTO UBUNTU

Once the tablet restarts you may see one of three screens:

1. A screen displays prompting you to start Ubuntu:
 - a. Hold down the power button until the tablet shuts down. Then, hold down the volume down button and the power button.

- b. When the Surface logo appears on the screen, release both. This should cause the device to default boot into Windows.

Note: This is necessary because upon a fresh boot, the keyboard doesn't always work, leaving you trapped at the Ubuntu startup screen.

2. A blue screen displays prompting you to choose an operating system:
 - a. Change the default to Windows 8.1.
 - b. Stay on this screen and scroll down to find the point at which to continue.
3. The device boots directly into Windows:
 - c. Log in normally.
 - d. Scroll down to find the appropriate course of action.

4.5.1. IF A BLUE SCREEN APPEARS ON STARTUP

1. Select Ubuntu from the screen. If this is successful, continue to Section 1.6. If not, continue to step 2.
2. Restart the device to get the blue screen to appear again.
3. Select Choose other options from the Change defaults or Choose other options, option at the bottom of the screen.
4. Go to the Use a device list and select Ubuntu. This should boot you into Ubuntu.

4.5.2. IF THE DEVICE BOOTS DIRECTLY INTO WINDOWS

1. Once you have logged in, swipe from the right side of the screen to open the charm.
2. Go to Settings > Change PC Settings > Update and recovery > Recovery > Advanced startup and click Restart now. A blue screen should appear when the device restarts.
3. Go to Use a device and select Ubuntu from the list. This should boot you into Ubuntu.

4.6. FIXING UBUNTU'S WI-FI PROBLEMS

Ubuntu has problems with the Surface Pro 2's Wi-Fi card. This section is dedicated to fixing those problems. If you are able to access the Wi-Fi from Ubuntu, ignore this section. If you are using the Pupil Pro, proceed to the appropriate sections of that tutorial. If not, continue with this tutorial.

1. Go to the following webpage, and download the driver. Rename it `usb8797_uapsta.bin` <http://jaxbot.me/articles/running-ubuntu-on-a-surface-pro-2-off-the-metal-video-5-26-14>
2. Copy the driver onto a flash drive, and plug that flash drive into the USB hub.
3. Type the following command into the terminal. If the file disappears from its original location, the process has worked.

```
sudo mv /driver/location /lib/firmware/mrvl
```

4. Restart the tablet to allow the driver to take effect. When you log back in, you should be able to access Wi-Fi in Ubuntu.

5. FINITE ELEMENT ANALYSIS USING ADINA SOFTWARE

This tutorial will give you a basic understanding of finite element analysis using ADINA software. This is a tutorial for those who are fairly new to finite element analysis, so you don't need to have any experience with the subject, although learning more about the topics we cover would be very helpful to you as you use ADINA. The finite element method itself is very mathematically rigorous, though we will aim to avoid the mathematics and focus primarily on the modeling process using ADINA, explaining advanced concepts in simple, understandable terms.

This tutorial requires that you have a copy of the ADINA software. We will not cover any specific practice problems here, so the free 900-node version of the software will work fine. Paper, pens, pencils, rulers, and other supplies are suggested; you will hopefully be drawing diagrams, creating charts, and writing down lots of information. This is very useful in the modeling process, as you will see later.

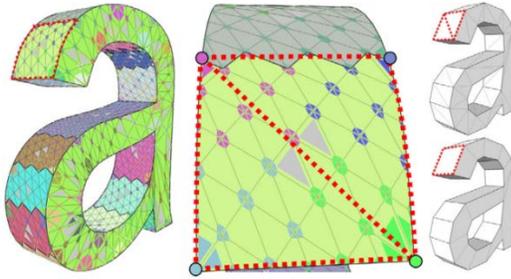
This tutorial will cover the following:

- Finite Element Analysis Overview
- Understanding the Physical Problem
- Creating a Finite Element Model of the Physical Problem
- Solving the Model / Post-Processing
- Interpreting Results and Refining the Model

5.1. FINITE ELEMENT ANALYSIS OVERVIEW

Finite element analysis (FEA) is a computational method used for modeling and analyzing complex physical systems. It gives users the ability to study in detail the state and properties of these modeled systems, which makes FEA a popular and powerful tool in engineering.

Finite element analysis works through approximation. The FEA software begins by breaking down an object into a collection of small pieces called **elements**, approximating the shape of the object, as has been roughly done with the letter a here:



To simplify the mathematics required to solve the problem, each of these elements is created to be regular in shape. Their individual behaviors are governed and predicted by systems of mathematical equations, which the computer easily calculates. Eventually, the computer adds their individual behaviors together to predict the behavior of the whole object.

Of course, before any of this can happen, the software requires a modeled object on which to break down and run computations. The software will need to know what the object's material is like, what loads may be on the object, and any other physical properties or constraints that would play into the solution of your problem. In order for you to construct a suitable model of the object and give the software all of this information, you will need to have a thorough understanding of the physical problem you're dealing with. The next section will give you an idea of what you should have prepared before you start modeling in ADINA.

5.2. UNDERSTANDING THE PHYSICAL PROBLEM

It is critical that you understand your problem thoroughly, and know all of the details that need to be considered during the modeling process. Proper documentation of these details is very useful to have, and it is strongly suggested that you have it prepared beforehand. This will make the modeling process smooth, efficient, and easy.

Draw out the problem in a detailed diagram on paper. Keep this with you throughout the modeling process. (Trust me, you'll need this.)

Drawing out the problem:

1. Draw the problem geometry. Define points, lines, surfaces, shapes, areas, volumes, axes, coordinate systems, etc.
 - a. All nodes, lines, and surfaces in ADINA must be labeled by numbers. It is recommended that you thoughtfully organize your numbering systems when labeling your geometry. Specifically and thoughtfully constructed numbering systems can assist in the efficiency of the solution, and they will also be useful as you go about constructing the geometry.
 - b. Nodes/points in ADINA are defined by coordinates in a coordinate system. You may want to create a chart defining numerical values of each node's coordinates that accurately describe the geometry.

Default Coordinate System: 0

Auto... Import... Export... Clear Del Row Ins Row

	Point #	X1	X2	X3	System
1	1	0.0	0.0	0.0	0
2	2	0.0	0.0	1.0	0
3	3	0.0	10.0	1.0	0
4	4	0.0	10.0	0.0	0
5					
6					

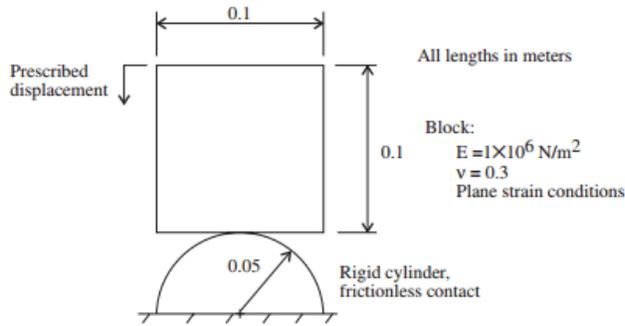


c. The following example shows a point definition chart in ADINA and the rectangle that it generates:

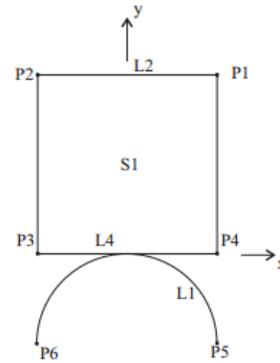
The rectangle is in the YZ-Plane (X2, X3). Pay close attention to the coordinate values and point numbers, and how they are organized in the generated rectangle.

2. Define boundaries and boundary conditions in your diagram.
3. Define any loads, displacements, forces, pressures, etc. along with their magnitudes.
4. Write down any relevant information. If there are any properties with the system or object that may be relevant to the solution of your problem, write them down somewhere near your diagram. Relevant information could include things such as the following:
 - Material Properties (e. g. Young's modulus, Poisson's ratio, density, etc.)
 - Moment of inertia
 - Contact conditions (e. g. friction between surfaces)
 - Constraint equations

Below is an example of a model plan taken from an [ADINA Practice Problem](#):

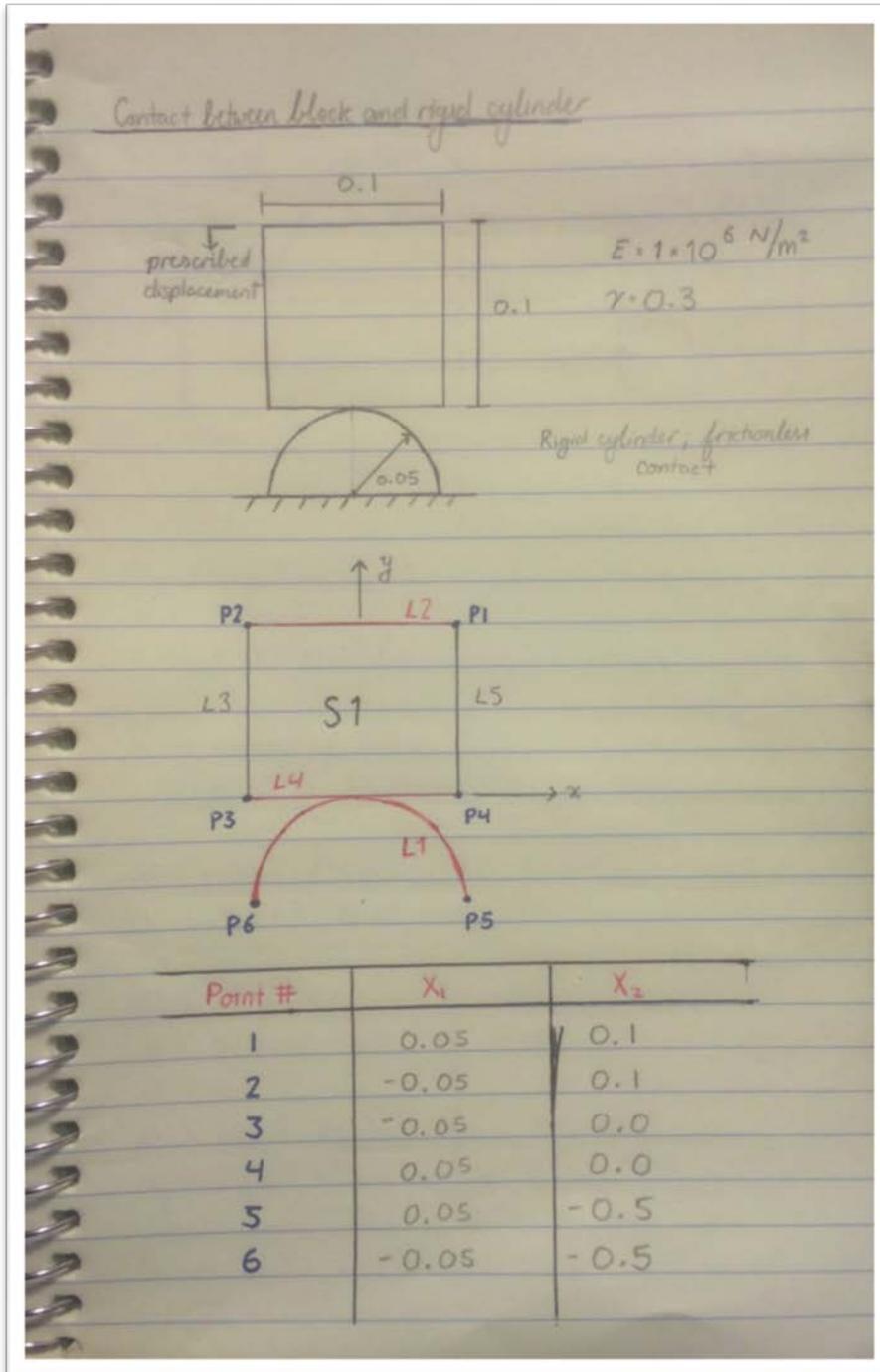


We would like to determine the displacements and stresses in the block when the block is pushed down 0.02 m.



A block in contact with a cylinder is shown. As you can see, the model geometry is sketched, measurements are described, material properties are defined (“ $E = \dots$ ” and “ $\nu = \dots$ ”), loads are described, a fixity is drawn under the half-cylinder (denoted by the slanted dashes), and a sketch defining points, lines, surfaces, and axes is provided.

On paper, it might look something like this:



Notice that I have included a chart laying out the coordinates of each point I want to define.

Ultimately, how you go about drawing, planning, organizing, and recording is up to you. As you become more familiar with the modeling process, you may feel that you are keeping track of too much or too little. Record whatever you feel is necessary or helpful to you when modeling.

5.3. CREATING A FINITE ELEMENT MODEL OF THE PHYSICAL PROBLEM

You may feel the inclination to build your model in a familiar software (e.g. SolidWorks) and import the model geometry into the ADINA AUI. This is certainly possible, but I would strongly recommend against doing this for a few reasons:

First, building your model in ADINA gives you complete control over and understanding of the specific geometry of the model. If you import a model, you have no say over how points are defined, how measurements are scaled, how lines or surfaces are defined, and so on. It will also be up to you to dig through the point/line/surface definition charts to figure out how the geometry has been laid out. Familiarity with the model geometry is crucial in describing other aspects of the problem.

Second, imports aren't always perfect, and they can be messy. In SolidWorks, for example, users must create various axes and planes to define where surfaces lie in space. When you import a SolidWorks model into ADINA, those axes appear as actual points and lines, which can be tricky to get rid of. In addition to this, curves in a model may also be represented as a series of several nodes in the shape of a curve, rather than being represented as a simple arc (which is a line type in ADINA). This can lead to incredibly complex and messy geometries, which will either be difficult or impossible to work with.

I will cover the process of modeling in the ADINA User Interface (AUI). With your notes and diagram at hand, open the AUI and make sure that ADINA Structures is selected in the Program Module drop-down list.

5.3.1. DEFINING MODEL CONTROL DATA

If you have decided that you want or need to configure model control settings, this will be your first step. Define problem headings, time functions, degrees of freedom, and whatever other control settings you would like to set before anything else. These can all be found under Control.

5.3.2. DEFINING MODEL GEOMETRY

Defining model geometry is a very important step in the modeling process. If you took notes, drew diagrams, made charts, or anything else, have those ready here.

To define points:

1. Find the Define Points icon,  or go to Geometry → Points...
2. Enter your node/point numbers and assign their coordinate values.

- a. X1, X2, and X3 correspond to X, Y, and Z, respectively.
- b. By default, 2-D solid models must be defined in the YZ-Plane (X2 and X3). The axes of the 2-D solid plane can be changed under Control → Miscellaneous Options.

To define lines:

1. Find the Define Lines icon,  or go to Geometry → Lines → Define...
2. Choose an appropriate line type from the Type: drop-down list.
3. Assign lines and line numbers accordingly for the geometry, taking node numbers, node positions, and planned surfaces into account.

To define surfaces:

1. Find the Define Surfaces icon,  or go to Geometry → Surfaces → Define...
2. Choose an appropriate surface type from the Type: drop-down list.
3. Define surfaces and assign surface numbers accordingly for the geometry.

Note: When defining the Vertex type surface, lines are drawn from P1 to P2, P2 to P3, P3 to P4, and P4 to P1. Keep this in mind. For triangles, use the same point for P1 and P4.

5.3.3. APPLYING LOADS AND BOUNDARY CONDITIONS

Once you have completed defining your model geometry, set the necessary loads and boundary conditions to be held throughout the solution. **Boundary conditions** are conditions that must be satisfied for a region of an object's boundary. For instance, a **fixity** is a boundary condition that requires a region of the boundary to stay fixed in position (fixities are going to be the most common type of boundary condition that you will use). Fixities can also be created to restrict movement only along specified axes/directions.

Applying Fixities:

1. Find the Apply Fixity icon  in the toolbar. You can also navigate to Model → "\Boundary Conditions → Apply Fixity.
2. Choose a geometric object type to apply fixities to such as points, lines, or surfaces.

Note: Changing the geometric object type will not delete any other fixities that you have entered for other geometric object types. The different types are kept separate from one another.

3. In the chart, enter the point/line/surface numbers that you would like to apply fixities to, and choose a fixity type for each one from the drop-down list in the next column. If you do not specify a fixity using the drop-down list in the chart, the default fixity will be applied. You can change or define the default fixity in the Default fixity section above the chart.
4. Click OK.

Loads can take the form of forces, displacements, pressures, velocities, accelerations, or any of a variety of types.

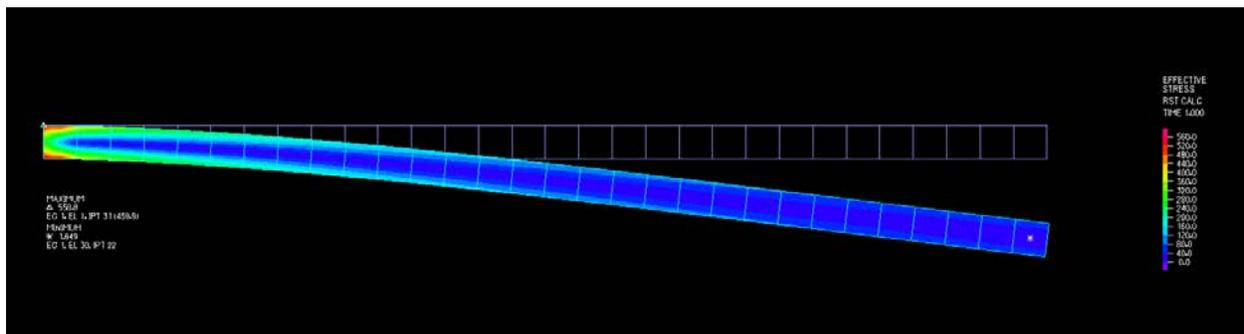
Applying Loads:

1. Find the Apply Load icon  in the toolbar. You can also navigate to Model → Loading → Apply....
2. Specify a load type from the Load Type drop-down list.
3. Click Define... to add a new load:
 - a. Enter the load's number, direction (as a vector), and magnitude.
 - b. If you want a force acting in the negative Z direction, enter [X: 0 | Y: 0 | Z: -1] under Force Direction.
 - c. Click OK.
4. Make sure the Load Number corresponds to the load you want to apply.
5. Choose a geometric object type to apply fixities to such as points, lines, or surfaces.
6. In the chart, enter the point/line/surface numbers that you would like to apply your load to.

Note: You can also choose time functions, arrival times, focus points, and other settings for the load here in this chart. Again, information for load types, numbers, and geometric object types are all kept separate from each other, just as with fixities.

7. Click Apply or OK.

In the example below, a fixity has been applied to the far left edge of a rectangular beam, while a downward-pushing force load has been applied to the far right end. You can see the mesh of the original, straight beam behind the deformed solution.



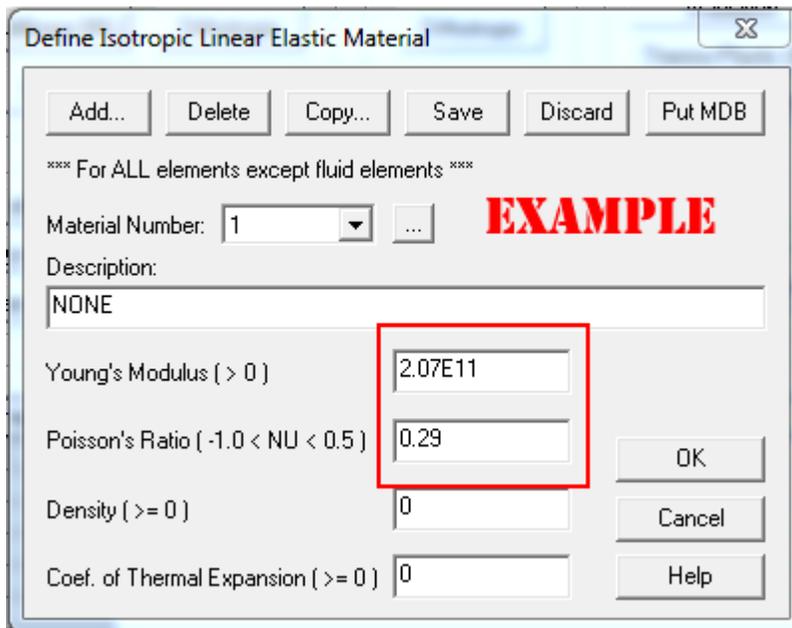
5.3.4. MANAGING MATERIAL DEFINITIONS

Materials are complex. There are a number of types, properties, settings, options, etc. that you can input for your materials in ADINA, and you can either construct a really complex, detailed material, or you can make a simple material defined by one or two properties. Unfortunately, if you want to understand all of the materials settings in ADINA, you will likely have to do quite a lot of studying and exploring on your own.

All of the materials settings can be accessed via the Manage Materials icon  on the toolbar, or can be found in the Manage Materials section in Model → Materials.

Below is an example using a simple isotropic elastic material with a Young's Modulus equal to 2.07×10^{11} and a Poisson's ratio equal to 0.29.

1. Go to the Manage Materials window, and select Isotropic under the Elastic section to open a new window to define an isotropic linear elastic material.
2. Add a new material, type *2.07E11* in the Young's Modulus box.
3. Type *0.29* in the Poisson's Ratio box and click OK.



If you want to create a specific rubber material, you will have to be familiar with one of the handful of material modeling methods you can use to do that, and you will have to supply the information necessary to model the material you want. As previously mentioned, this tutorial is not the place to cover all of the innumerable aspects of modeling various kinds of materials. It will be primarily your job to figure out what you need to do to achieve the goals you have in mind for your materials.

5.3.5. MESHING

The geometry that you defined is like the frame of your model. A mesh gives that frame substance and a dynamic, structured body. For your mesh to work properly for your model, you need to choose appropriate settings for mesh density, point size, and so on. But before you get started with that, you need to define an Element Group.

Define an element group:

1. Find the Define Element Groups icon  on the toolbar, or navigate to Meshing → Element Groups....
2. Once there, add a group, select an appropriate type for your model, and if you want, adjust any necessary settings in the below tabs.
3. Click OK.

The procedure below describes one of many ways to choose mesh settings.

Choose mesh settings:

1. Go to Mesh Density and select Complete Model....
2. Make sure the Subdivision Mode is set to Use End-Point Sizes and click OK.
3. Go back to Mesh Density and select Point Size.
4. Choose All Geometry Points from the Points Defined from: drop-down list, and set a maximum mesh size.

Note: Setting your maximum mesh size to 1 will set a mesh point every 1 unit of length along your geometry, setting it to 2 will set a mesh point every 2 units of length, and so on.

5. Click Apply or OK.

Once you are done choosing your mesh settings, generate the mesh on the point, line, surface, or volume that you want. Select the Mesh Points/Lines/Surfaces icons on the toolbar or navigate to Meshing, open the Create Mesh menu, and select a geometric object type.

If you need to refine or adjust your mesh, you can delete it using the Delete Mesh icon  on the tool bar or by selecting Delete Mesh... under Meshing. Adjust your mesh settings as you see fit, and try it again until you have a proper mesh.

Once you finish this, you should be ready to solve your model!

5.4. SOLVING THE MODEL / POST-PROCESSING

So you have finished building your model, and now it's time to do some magic. You are going to save the file containing your model, and then run the solution. This part is probably the quickest, but this is where the heart of the ADINA software is.

To solve the model:

1. Create a file name and save your model as an .idb file (e.g. *filename.idb*).
2. Find the Data File/Solution icon  on the toolbar, or navigate to Solution and select Data File/Run...:
 - a. A window will display, prompting you to enter a filename for a .dat file. Name the file to match that of your .idb file. (e.g., *filename.dat*)

- b. Check the Run Solution box near the bottom left of the window.
 - c. Click Save.
3. Several windows will display. Resolve any errors and close the windows.

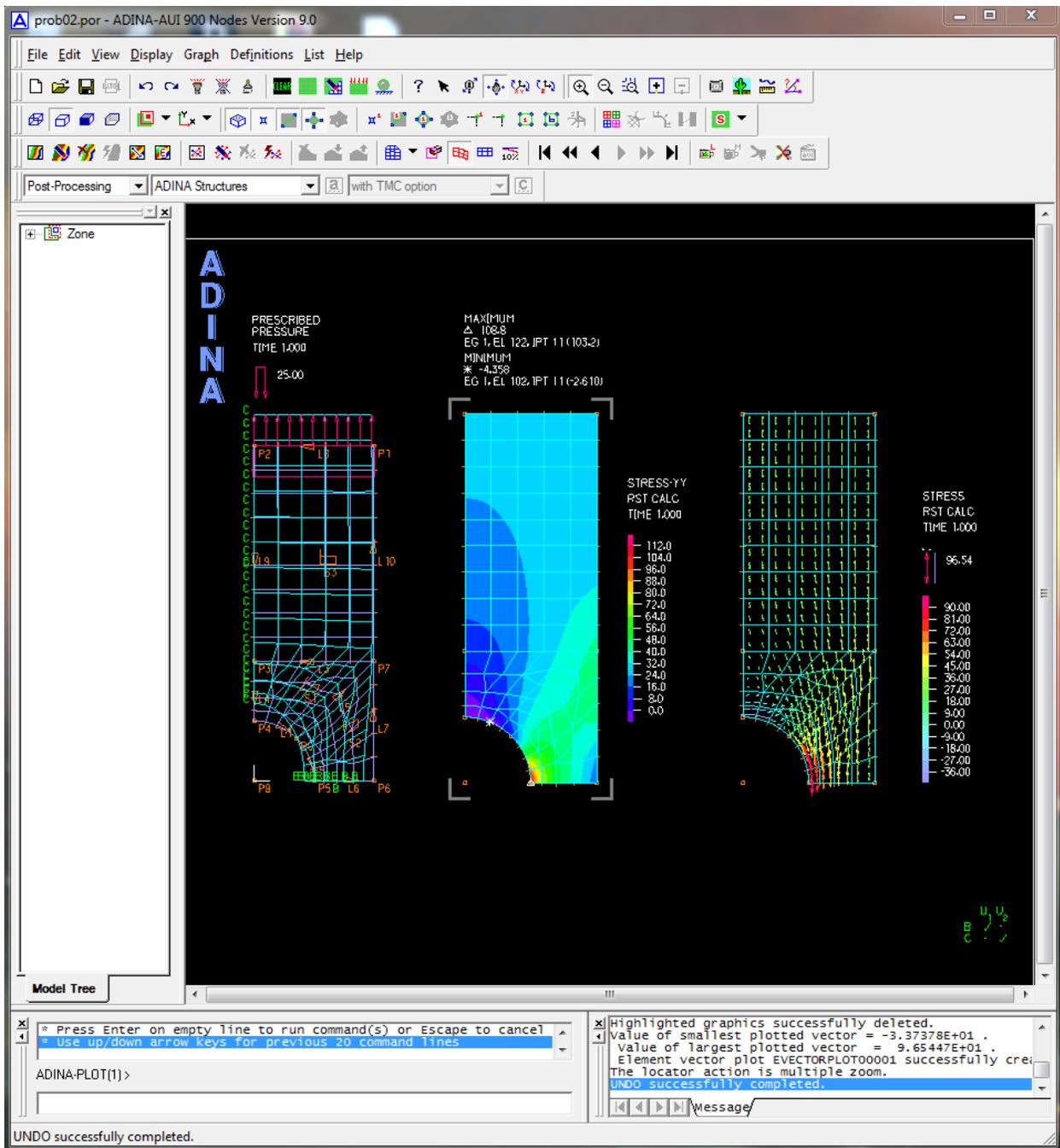
To view your solution:

1. Go to the Program Module drop-down list. ADINA Structures should be the current selection.
2. Open the list and select Post-Processing. Feel free to discard all changes (you just saved the file, so it'll be perfectly fine). You should have a blank slate in front of you.
3. Open your .por file that was generated by the solution process. The filename will match the .dat filename. You should see what looks like your original mesh.

At this point, you can do fun things with your solution:

- You can show the deformations made by the load by finding and selecting the Scale Displacements icon  10% in the toolbar.
- If you want to see your original mesh for comparison, you can generate a copy of it by selecting the Show Original Mesh icon.
- You can also generate another mesh plot, which will be a copy of what you started with when you opened the .por file.
- You can generate band plots and vector plots of different variables. These can be generated by selecting the Create Band/Vector Plot icons  on the toolbar, or by navigating to Display, opening the appropriate plot menu, and clicking Create....

The below example contains a deformation comparison, a stress band plot, and a stress vector plot, respectively. These were all done in one Post-Processing session. You can generate as many of these as you like in a single window.



If you are an experienced user, you can also create graphs and animations in Post-Processing if you define appropriate time functions during the modeling process. You can display lists of the collected data, export those lists of data as .txt files, take screenshots of your solution, and export animations of systems as .avi files. Play around with the various tools until you feel the data is well represented.

5.5. INTERPRETING THE RESULTS AND REFINING THE MODEL

You have finished solving your problem. So, what does it look like? What does it mean? What are the implications? What can you learn from the data generated? Does your solution look the way you expected? Is there anything wrong?

Learn as much as you can from the solution (be sure to save the file). Record everything that you can learn and try to record as much data as you can. A lot of useful information is given. If you need to, you can refine your model and run the solution process again.

To access your model for revision, select “ADINA Structures” from the Program Module drop-down list (feel free to discard changes) and open your *filename.idb* file. Make any necessary revisions and follow the steps in the previous section (Section 1.4.) to run the solution process again. You can give your solution file a different name if you want to keep your first one.

I recommend giving a go at some of the practice problems here: <http://www.adina.com/tutorials.shtml>

Some of them may require the full version of ADINA, so if you are using the 900-nodes version, be sure to verify that you can solve the problem with it.

6. LOADING UBUNTU ONTO A FLASH DRIVE AND USING PUPIL LABS SOFTWARE

This tutorial documents how to set up Ubuntu on a flash drive and set up software created by Pupil Labs to record and play back eye tracking data.

The materials necessary for this tutorial include the following items:

- Laptop computer
- Pupil Pro headset
- Flash drive at least 4 GB in size

This tutorial includes the following topics

- Setting up Ubuntu
- Installing Pupil Capture
- Installing Pupil Player
- Using Pupil Capture and Pupil Player

6.1. SETTING UP UBUNTU

Download Ubuntu desktop from the following website:

<http://www.ubuntu.com/download>

Next, download Universal USB Installer from this website: <http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/>

1. Plug the flash drive into the laptop and follow the instructions on the website above to install Ubuntu onto the flash drive. Be sure to leave at least 2000 MB of persistence so that the downloaded files will be saved.
2. Once Ubuntu has been installed, restart the computer. Press F12 during the boot to open the boot list. Select the USB Drive from the list and hit Enter.
3. When the Desktop displays Ubuntu, it will prompt you to choose to install or try Ubuntu. Select Try Ubuntu.
4. Connect to Wi-Fi by clicking on the Wi-Fi symbol on the bar at the top of the screen, in the upper right-hand corner.

6.2. INSTALLING PUPIL CAPTURE

Download the latest Pupil Capture release from the following website:

<https://github.com/pupil-labs/pupil/releases/>

1. Open the zip file and extract the contents to the desktop.
2. Connect the two cameras in the Pupil Pro headset to the computer via USB.
3. Open the Pupil Capture folder and find the .exe file.
4. Double click the .exe file to start the program. If two windows open, one for each of the two cameras, then the program is working correctly.
5. Close Pupil Capture.

6.3. INSTALLING PUPIL PLAYER

1. Download the zip file from this website:
<https://github.com/pupil-labs/pupil>
2. Extract the pupil_master folder to the desktop.
3. Open the Ubuntu Software Center from its icon in the Launch Bar on the left side of the screen. Its icon should be an A directly above the Amazon icon.
4. Go to the Software Center's menu by moving the mouse to the top bar while Software Center is selected.
5. Click Edit.
6. Click Software Sources.
7. Check the box next to Community-maintained free and open-source software (universe).
8. Go to the search bar in the software center and search Fortran.
9. Locate the GNU Fortran 95 Compiler and install it.
10. Select the icon at the top of the Ubuntu Launcher, and search for the terminal. Type in or copy and paste in the following commands in the order they're listed.

Note: Copying and pasting may cause a 1. to appear at the start of the command. Make sure to delete that.

Note: If a line is indented, that means it is a continuation of the previous line.

```
sudo apt-get update
sudo apt-get install python-opengl mesa-common-dev libglu1-mesa-dev git python-setuptools
libv4l-dev v4l-utils cmake python-zmq python-pip libav-tools python-opencv
```

11. Copy and paste in the following terminal commands one at a time. Each line is a separate command. Type them in the order that they are given.

```
sudo apt-get install python-dev
sudo pip install numexpr
sudo apt-get install python-numpy python-scipy python-matplotlib
sudo apt-get install xorg-dev
```

12. Copy and paste in the following commands to install anttweakbar.

```
cd ~/
wget -O AntTweakBar_116.zip
http://sourceforge.net/projects/anttweakbar/files/latest/download?source=dlp
unzip AntTweakBar_116.zip
cd AntTweakBar/src
make
sudo mv ../lib/libAntTweakBar.so /usr/lib/
sudo mv ../lib/libAntTweakBar.so.1 /usr/lib/
cd ~/
rm -rf AntTweakBar_116.zip
rm -rf AntTweakBar
```

13. Copy and paste in the following commands to install glfw3.

```
cd ~/
git clone http://github.com/glfw/glfw
cd glfw/
cmake -G "Unix Makefiles" -DBUILD_SHARED_LIBS=TRUE
sudo make install
sudo ln -s /usr/local/lib/libglfw.so.3 /usr/lib/libglfw.so.3
cd ../
sudo rm -r glfw
```

6.4. USING PUPIL CAPTURE AND PUPIL PLAYER

1. Run Pupil Capture, and move your head so that the entire screen is within the view of the world camera.
2. To calibrate the camera, press the C key on the keyboard, and follow the target around the screen with your eyes.
3. Verify that a red dot is appearing on the world camera window, showing where you are looking.
4. The R key will start and stop a recording. Create a recording to test in Pupil Player. If Pupil Capture crashes upon ending the recording, the file has been saved and exit out of the program.
5. Open the terminal and use the following commands. For the parts of the commands in quotation marks, you can find the folder you are looking for in the GUI, and drag the folder onto the terminal. The recordings are stored under Pupil Capture, in the recordings folder. The directory is the folder with the triple digit name.

```
cd "path_to_pupil_dir/pupil_src/player"
python main.py "path/to/recording_directory"
```

6. Press the space bar to start the playback of the recording.

6.5. PATCH TO RUN PUPIL CAPTURE OFF OF A USB HUB

In the case where there are not enough USB ports to plug in both cameras and any necessary USB devices, a USB hub is necessary. For example, if the Pupil Pro is connected to a Surface Pro 2 tablet, a hub will be necessary. Only use this section if necessary.

Note: If you are using a hub, you will need at least a 16 GB flash drive.

Open the terminal and type in the following commands:

```
sudo apt-get install patchutils libproc-processtable-perl
git clone git://linuxtv.org/media_build.git
cd media_build
./build --main-git
cd media
wget
https://gist.githubusercontent.com/mkassner/10134241/raw/1c9f632bd1894fb7dab000ba4dbcadcbf
1ca271f/mjpeg_bandwidth.patch
git apply mjpeg_bandwidth.patch
cd ../
make -C v4l/
sudo make install
sudo rmmod uvcvideo
sudo modprobe uvcvideo quirks=0x440 jpg_compression=2
sudo sh -c 'echo "options uvcvideo quirks=0x440 jpg_compression=2" >
/etc/modprobe.d/pupil_uvc_cam.conf'
```

Certain commands will take a long time, as they are downloading and installing big packages. Ignore any delays or non-critical errors.

Note: The command starting with 'wget' spans three lines, and the last command spans two lines.

Note: The 'sudo modprobe' command may give an error. If it does, ignore it and continue.