Calculating the Stresses in a Thick-Walled Pressure Cylinder and Sphere Using JavaScript

Zachary Barnett

MAE 4344-001: Computer Aided Engineering

Honors Project

Abstract

Light-weight and fast engineering computational utilities can be made using JavaScript, a programming language that works on modern web browsers. In this project, a small application was developed using JavaScript. The goal of the application was to take user input, allow the user to select measurement units, and calculate and display the stress generated in a thick-wall pressure vessel. HTML and CSS languages, used to display the majority of internet pages, was also used. HTML created the structure of the page, while CSS was used to style the page and provide a figure of the pressure vessel. JavaScript was chosen due to the ability of the browser to perform calculations on the user side; in other words, the browser does not have to deliver and receive the calculations to a server in a different location. The developed application does exemplify the speed and ease of calculations by JavaScript, calculating the stress results instantaneously from the user's perspective.

Background

HTML, CSS, and JavaScript all work together for the purpose of taking the user input in a styled form and calculating results to display. HTML specifically forms the structure of the page, and is used to create the individual components such as the text field boxes in which the user types the numbers. CSS is used to organize the page, and style the page elements such as figures. JavaScript, responsible for the main calculations, also changes the HTML and CSS page elements and styles based on the user choice. [1] In this application, the user chooses between calculating the results for a sphere or a cylinder pressure vessel.

For finding the stress in a cylinder or spherical thick-wall pressure vessel, the established equations were taken from resources and applied within the programming. In this way, the application is intended to quickly use the equations to get the stress results the user wants. These equations are programmatically simple in that basic operations are used. The equations are already solved through Calculus and Algebra for the desired variable and therefore the developed application can substitute input for the variables [2].

This application calculates using thick-walled equations, meaning that a thin-wall assumption cannot be made. (A thin-wall assumes that the thickness of the wall is less than one tenth of its inner radius.)

The equations for stress are given below, starting with the cylinder [3]:

$$\text{Axial Stress: } \sigma_a = \frac{P_i r_i^2 - P_o r_o^2}{r_o^2 - r_i^2}$$

$$\text{Circumferential (Hoop) Stress: } \sigma_h = \frac{P_i r_i^2 - P_o r_o^2}{r_o^2 - r_i^2} - \frac{r_i^2 r_o^2 (P_o - P_i)}{r^2 (r_o^2 - r_i^2)}$$

$$\text{Radial Stress: } \sigma_r = \frac{P_i r_i^2 - P_o r_o^2}{r_o^2 - r_i^2} + \frac{r_i^2 r_o^2 (P_o - P_i)}{r^2 (r_o^2 - r_i^2)}$$

For a sphere, the equations are slightly more complicated:

$$\text{Radial Stress: } \sigma_r = \frac{P_o r_o^3}{r^3}\left(\frac{r^3 - r_i^3}{r_i^3 - r_o^3}\right) + \frac{P_i r_i^3}{r^3}\left(\frac{r_o^3 - r^3}{r_i^3 - r_o^3}\right)$$

$$\text{Circumferential (Hoop) Stress: } \sigma_h = \frac{P_o r_o^3}{2r^3}\left(\frac{2r^3 - r_i^3}{r_i^3 - r_o^3}\right) - \frac{P_i r_i^3}{2r^3}\left(\frac{2r^3 + r_o^3}{r_i^3 - r_o^3}\right)$$

Where:

$$P_i = Internal\,Pressure$$

$$r_i = Internal\,Radius$$

$$P_o = Internal\,Pressure$$

$$r_o = Internal\,Radius$$

$$r = Radius\,¿\,point\,within\,wall\,,at\,which\,point\,the\,stress\,is\,found$$

In Figure 1 below, these variables are shown visually. While *r* is not shown, it can be imagined as the distance to a point within the wall.

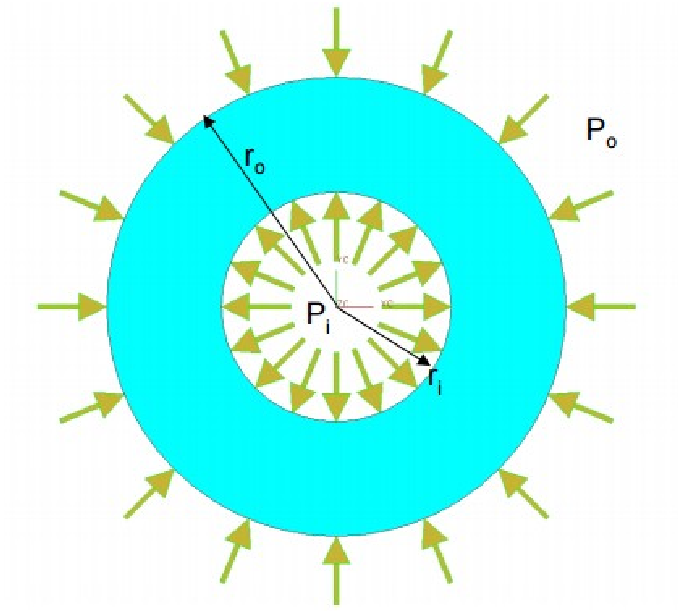


Figure 1: Variables Used in Calculations

With JavaScript code, the variables can be checked for obvious errors. For instance, the user can be prompted and the calculation cancelled if the user accidentally enters a negative radius, or an *r* that does not lie within the thickness of the wall.

Program Flow

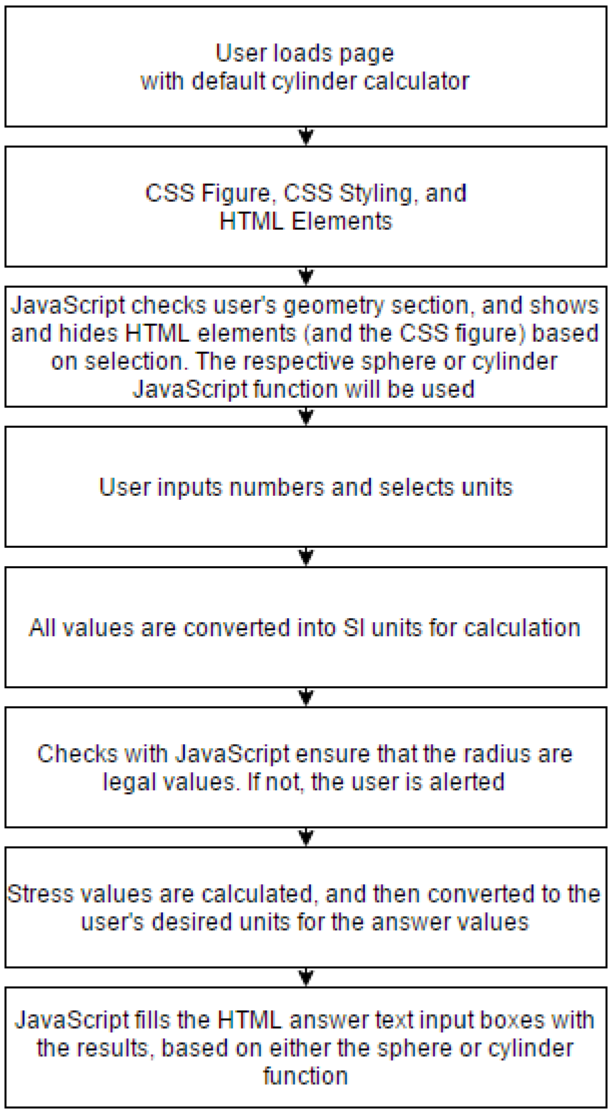
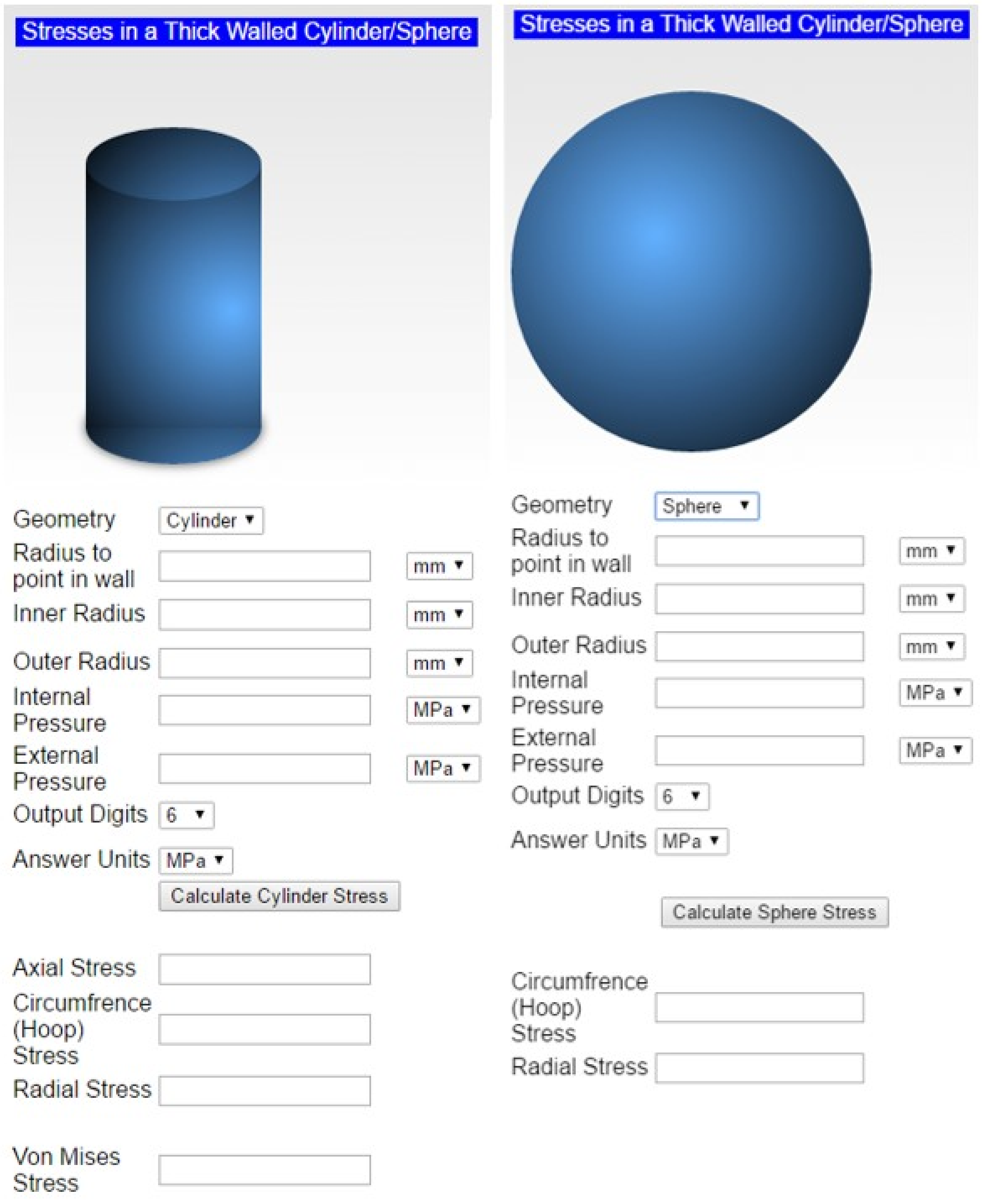

User loads page with default cylinder calculator
CSS Figure, CSS Styling, and HTML Elements
JavaScript checks user's geometry section, and shows and hides HTML elements (and the CSS figure) based on selection. The respective sphere or cylinder JavaScript function will be used
User inputs numbers and selects units
All values are converted into SI units for calculation
Checks with JavaScript ensure that the radius are legal values. If not, the user is alerted
Stress values are calculated, and then converted to the user's desired units for the answer values
JavaScript fills the HTML answer text input boxes with the results, based on either the sphere or cylinder function

Figure 2: Program Flow Chart

Upon loading the page, the user sees the default cylinder stress calculator. HTML elements such as drop down boxes and text input fields allow the user to select and input information. A cylinder figure is drawn using a combination of CSS and HTML, with the HTML controlling the location on the page and the CSS controlling the look of the figure. At this point, the user can select the geometry. If sphere is selected in the geometry dropdown box, a JavaScript function will change the page by showing and hiding certain HTML elements. In Figure 3, a screenshot of the browser shows what the user sees upon loading the page and either selecting cylinder or sphere in the geometry drop down box.


Stresses in a Thick Walled Cylinder/Sphere

Geometry: Cylinder
Radius to point in wall: mm
Inner Radius: mm
Outer Radius: mm
Internal Pressure: MPa
External Pressure: MPa
Output Digits: 6
Answer Units: MPa

Stresses in a Thick Walled Cylinder/Sphere

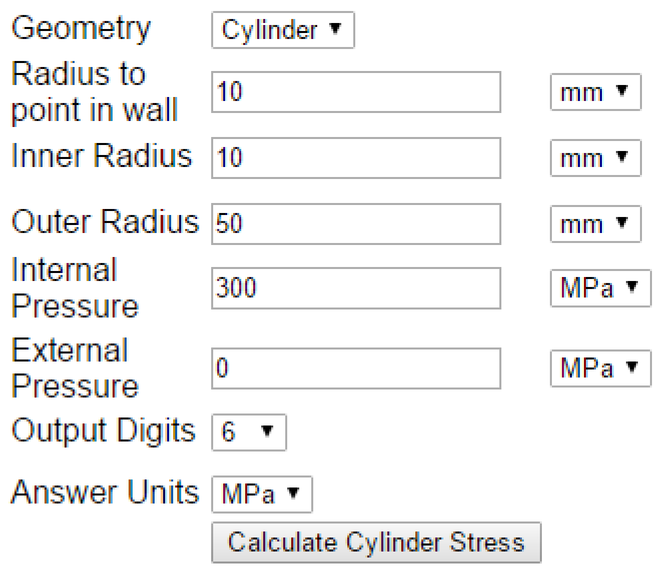
Geometry: Sphere
Radius to point in wall: mm
Inner Radius: mm
Outer Radius: mm
Internal Pressure: MPa
External Pressure: MPa
Output Digits: 6
Answer Units: MPa

Figure 3: Left: Application with Cylinder Geometry Selected, Right: Sphere Geometry Selected

The user can then input the numbers. For each number, the user can select a specific unit, and each JavaScript function for sphere and cylinder calculation support mixed units. The user can also change the precision of the answer (output digits) and also select the stress answer unit. Once all the number are entered, the user clicks the calculate button, and the respective cylinder or sphere answers are shown. In order for this to occur, the JavaScript function takes the input in the select boxes and input boxes and converts them to JavaScript variables. JavaScript converts the measurement to SI units, and then performs checks. For example, a check is performed so that the radius to the point in the wall thickness must lie within the wall, the inner radius must be smaller than the outer radius, and radius values cannot be negative. If any of these are detected as problems, the user is alerted and the program stops before giving the user erroneous numbers. Otherwise, the respective JavaScript function substitutes the SI values into the stress equations. Before the function outputs the result in the HTML text boxes designated for the answers, the values are converted to the answer units, based on the user's choice.

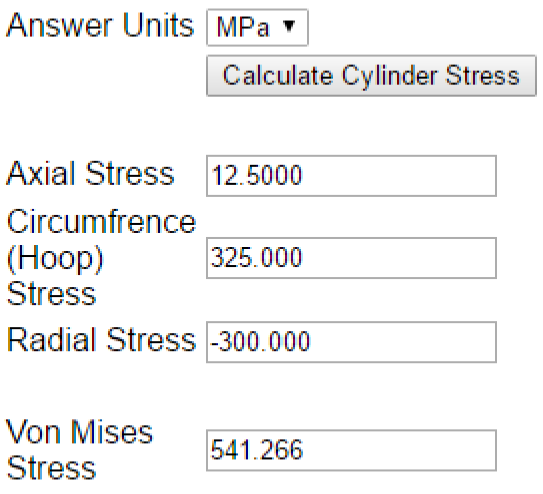
Example Run

In this example, a thick-walled cylinder has an inner radius of 10mm, an outer radius of 50mm, with external pressure of 0 MPa and inside pressure 300 MPa. Figure 4 shows the setup, the default settings for precision, and the change of the input and output units accordingly. The radius to the stress element in the wall will be varied at 10 mm, 25 mm, and 50 mm. An incorrect radius of 5 mm will also be used to show error checking.



Geometry	Cylinder ▼	
Radius to point in wall	10	mm ▼
Inner Radius	10	mm ▼
Outer Radius	50	mm ▼
Internal Pressure	300	MPa ▼
External Pressure	0	MPa ▼
Output Digits	6 ▼	
Answer Units	MPa ▼	

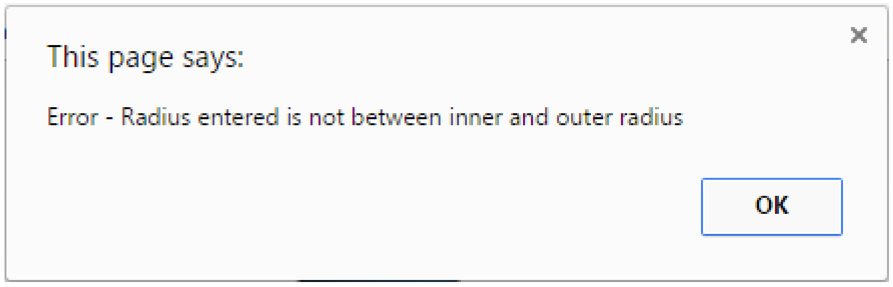
Figure 4: Setup with 10mm radius to wall element



Axial Stress	12.5000
Circumfrence (Hoop) Stress	325.000
Radial Stress	-300.000
Von Mises Stress	541.266

Figure 5: Output

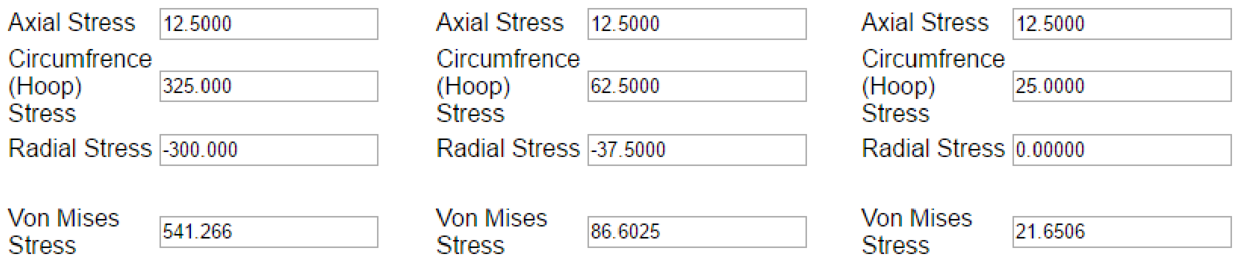
As an error check, 5 mm was entered into the "radius to point in wall" field. As expected, the program displayed an error. Figure 6 shows the resulting error message.


This page says:

Error - Radius entered is not between inner and outer radius

Figure 6: Error message when using 5 mm radius

Now that the application is demonstrated to work and also display an error case, the results of varying the radius (10 mm, 25 mm, 50 mm) are given in Figure 7.



Axial Stress	12.5000	Axial Stress	12.5000	Axial Stress	12.5000
Circumfrence (Hoop) Stress	325.000	Circumfrence (Hoop) Stress	62.5000	Circumfrence (Hoop) Stress	25.0000
Radial Stress	-300.000	Radial Stress	-37.5000	Radial Stress	0.00000
Von Mises Stress	541.266	Von Mises Stress	86.6025	Von Mises Stress	21.6506

Figure 7: Error message when using 5 mm radius

These results match a separate reference, in which the answers have been found before. The values given by the JavaScript application match those given by the reference. [4] Although this was a quick test, it shows that this application has the ability to almost instantaeneously solve thick-walled cylinder stress problems.

Conclusion

The thick-walled cylinder and sphere stress calculator application was developed and worked as expected. By entering the radii and pressures, the stress at a certain point in the wall thickness is calculated by the light-weight JavaScript application. This is just one example that shows the possibilities of coding applications within JavaScript. These applications can be easily distributed and save many users time in similar engineering problems.

References

[1] Lawrence, K. L., B. H. Dennis, and B. P. Wang. *Light-Weight, Client-Side Engineering Computational Utilities*. Print.

[2] https://en.wikipedia.org/wiki/Cylinder_stress

[3] Norton, Robert L. *Machine Design: An Integrated Approach*. Upper Saddle River, NJ: Pearson Prentice Hall, 2006. Print.

[4] http://www2.mae.ufl.edu/haftka/adv-elast/lectures/Sections11.1-3.pdf

Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="en-us" http-equiv="Content-Language" />
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Stress in Thick Walled Cylinder or Sphere</title>
<style type="text/css">
.auto-style2 {
font-family: Arial, Helvetica, sans-serif;
}
.auto-style3 {
color: #FFFFFF;
text-align: center;
font-family: Arial, Helvetica, sans-serif;
font-size: medium;
}
/*Sphere must be 250px or form will collide with drawing*/
.sphereDraw {
display: block;
background: black;
margin: 0;
border-radius: 50%;
height: 250px;
```

```
width: 250px;
background: radial-gradient(circle at 100px 100px, #5cabff, #000);
}
.sphereDraw .shadow {
position: absolute;
width: 100%;
height: 100%;
background: radial-gradient(circle at 50% 50%, rgba(0, 0, 0, 0.4), rgba(0, 0,
0, 0.1) 40%, rgba(0, 0, 0, 0) 50%);
-webkit-transform: rotateX(90deg) translateZ(-150px);
-moz-transform: rotateX(90deg) translateZ(-150px);
-ms-transform: rotateX(90deg) translateZ(-150px);
-o-transform: rotateX(90deg) translateZ(-150px);
transform: rotateX(90deg) translateZ(-150px);
z-index: -1;
}
.cylDraw {
position: absolute;
margin:50px;
}
.cylDraw .middle{
width:120px;
height:180px;
background:radial-gradient(circle at 100px 100px, #5cabff, #000);
position:absolute;
}
.cylDraw .top{
width: 120px;
height: 50px;
background:radial-gradient(circle at 100px 100px, #5cabff, #000);
-moz-border-radius: 60px / 25px;
-webkit-border-radius: 60px / 25px;
border-radius: 60px / 25px;
position:absolute;
top:-25px;
}
.cylDraw .bottom{
width: 120px;
height: 50px;
background:radial-gradient(circle at 100px 100px, #5cabff, #000);
-moz-border-radius: 60px / 25px;
-webkit-border-radius: 60px / 25px;
border-radius: 60px / 25px;
```

```
position:absolute;
top:155px;
box-shadow:0px 0px 10px rgba(0,0,0,0.75)
}
.stage {
width: 300px;
height: 300px;
-webkit-perspective: 1200px;
-moz-perspective: 1200px;
-ms-perspective: 1200px;
-o-perspective: 1200px;
perspective: 1200px;
-webkit-perspective-origin: 50% 50%;
-moz-perspective-origin: 50% 50%;
-ms-perspective-origin: 50% 50%;
-o-perspective-origin: 50% 50%;
perspective-origin: 50% 50%;
}
body {
width: 300px;
margin: 20px auto;
background: linear-gradient(to bottom, rgba(100, 100, 100, 0.2) 0%, rgba(255,
255, 255, 0.5) 40%, #ffffff 100%);
background-repeat: no-repeat;
}
</style>
</head>
<body>
<div id="formInformation">
<form id=formCyl>
<table style="width: 100%">
<tr>
<td>
<table style="width: 100%">
<tr>
<td>
<table style="width: 100%">
<tr>
<td style="background-color:blue;" class="auto-style3"> Stresses in a Thick
Walled Cylinder/Sphere</td>
</tr>
<tr>
<td>&nbsp;</td>
```

```
</tr>
<!--
<tr id="cylinfo">
<td class="auto-style2">A cylinder with closed ends will have an axial stress
in the walls.</td>
</tr>
-->
</table>
</td>
</tr>
<table id="cylT">
<tr>
<td>
<figure class="cylDraw">
<section class="bottom"></section>
<section class="middle"></section>
<section class="top"></section>
</figure>
</td>
</tr>
</table>
<table id="sphereT" style="visibility:hidden">
<tr>
<td>
<figure class="sphereDraw" ><span class="shadow"></span>
</figure>
</td>
</tr>
</table>
<tr>
<td>
<table style="width: 100%">
<tr>
<td style="width: 33%" class="auto-style2">Geometry</td>
<td id ="geom" style="width: 33%">
<select onchange=showHide(); id="geo">
<option value="0" selected>Cylinder</option>
<option value="1">Sphere</option>
</select>
</td>
</tr>
<tr>
<td style="width: 35%" class="auto-style2">Radius to point in wall</td>
```

```
<td style="width: 35%"><INPUT value="" maxLength=9 size=16 name=r> </td>
<td style="width: 35%"><select id="ropt">
<option value="0" selected>mm</option>
<option value="1">cm</option>
<option value="2">m</option>
<option value="3">in</option>
<option value="4">feet</option>
</select>
</td>
</tr>
<tr>
<td style="width: 33%" class="auto-style2">Inner Radius</td>
<td style="width: 33%"><INPUT value="" maxLength=9 size=16 name=innr> </td>
<td style="width: 33%"><select id="innro">
<option value="0" selected>mm</option>
<option value="1">cm</option>
<option value="2">m</option>
<option value="3">in</option>
<option value="4">feet</option>
</select>
</td>
</tr>
<tr>&nbsp;</tr><tr>&nbsp;</tr><tr>&nbsp;</tr><tr>&nbsp;</tr>
<tr>
<td style="width: 33%" class="auto-style2">Outer Radius</td>
<td style="width: 33%"> <INPUT value="" maxLength=9 size=16 name=outr></td>
<td style="width: 33%"><select id="outro">
<option value="0" selected>mm</option>
<option value="1">cm</option>
<option value="2">m</option>
<option value="3">in</option>
<option value="4">feet</option>
</select>
</td>
</tr>
<tr>
<td style="width: 33%" class="auto-style2">Internal Pressure</td>
<td style="width: 33%"> <INPUT value="" maxLength=9 size=16 name=inp></td>
<td style="width: 33%"><select id="innpo">
<option value="0" selected> MPa </option>
<option value="1"> kPa </option>
<option value="2"> Pa </option>
```

```
<option value="3"> psi </option>
<option value="4"> ksi </option>
<option value="5"> Msi </option>
</select>
</td>
</tr>
<tr>
<td style="width: 33%" class="auto-style2">External Pressure</td>
<td style="width: 33%"> <INPUT value="" maxLength=9 size=16 name=outp></td>
<td style="width: 33%"><select id="outpo">
<option value="0" selected> MPa </option>
<option value="1"> kPa </option>
<option value="2"> Pa </option>
<option value="3"> psi </option>
<option value="4"> ksi </option>
<option value="5"> Msi </option>
</select>
</td>
</tr>
<tr>
<td style="width: 33%" class="auto-style2">Output Digits </td>
<td style="width: 33%"> <select id="p">
<option value="0"> 1 </option>
<option value="1"> 2 </option>
<option value="2"> 3 </option>
<option value="3"> 4 </option>
<option value="4"> 5 </option>
<option value="5" selected> 6 </option>
<option value="6"> 7 </option>
<option value="7"> 8 </option>
<option value="8"> 9 </option>
<option value="9"> 10 </option>
<option value="10"> 11 </option>
<option value="11"> 12 </option>
<option value="12"> 13 </option>
<option value="13"> 14 </option>
<option value="14"> 15 </option>
</select></td>
</tr>
<tr>&nbsp;</tr>
<tr>&nbsp;</tr>
<tr>&nbsp;</tr>
```

```
<tr>&nbsp;</tr>
<tr>
<td style="width: 33%" class="auto-style2">Answer Units</td>
<td id ="cylAnswer" style="width: 33%"> <select id="answerUnits">
<option value="0" selected> MPa </option>
<option value="1"> kPa </option>
<option value="2"> Pa </option>
<option value="3"> psi </option>
<option value="4"> ksi </option>
<option value="5"> Msi </option>
</select>
</td>
</tr>
<tr>
<td style="width: 35%">&nbsp;</td>
<td id = "cylButton" class="auto-style3" ><INPUT type = "button" value=
"Calculate Cylinder Stress" onclick=compute(form); maxLength=7
size=16 ></td>
</tr>
<tr>
<td style="width: 35%">&nbsp;</td>
<td id = "sphereButton" class="auto-style3" style="visibility:hidden"><INPUT
type = "button" value= "Calculate Sphere Stress" onclick=computeSphere(form);
maxLength=7
size=16 ></td>
</tr>
<tr id = "axialStress">
<td style="width: 33%" class="auto-style2">Axial Stress</td>
<td style="width: 33%"><INPUT value= " " maxLength=7 size=16
name=astress></td>
<td style="width: 33%">&nbsp;</td>
</tr>
<tr>
<td style="width: 33%" class="auto-style2">Circumfrence (Hoop) Stress</td>
<td style="width: 33%"><INPUT value= " " maxLength=7 size=16
name=cstress></td>
<td style="width: 33%">&nbsp;</td>
</tr>
<tr>
<td style="width: 33%" class="auto-style2">Radial Stress</td>
<td style="width: 33%"><INPUT value= " " maxLength=7 size=16
name=rstress></td>
<td style="width: 33%">&nbsp;</td>
</tr>
<td style="width: 33%">&nbsp;</td>
```

```
<tr id = "cylVonMises">
<td style="width: 33%" class="auto-style2">Von Mises Stress</td>
<td style="width: 33%"><INPUT value= " " maxLength=7 size=16
name=vstress></td>
</tr>
<tr>
<td style="width: 33%">&nbsp;</td>
<td style="width: 33%">&nbsp;</td>
<td style="width: 33%">&nbsp;</td>
</tr>
</table>
</td>
</tr>
</table>
</td>
</tr>
</table>
</form>
</div>
</body>
<SCRIPT language=javascript type=text/javascript>
//assumes default will always be cylinder
//document.getElementById('geo').selectedIndex = 0;
var shape = "cylinder"
var cylH = document.getElementById('geo');
cylH.selectedIndex = 0;
function showHide() {
if (shape == "cylinder") {
shape = "sphere";
document.getElementById("cylButton").style.visibility="collapse";
document.getElementById("cylVonMises").style.visibility="collapse";
document.getElementById("sphereButton").style.visibility="visible";
document.getElementById("axialStress").style.visibility="collapse";
//drawing
document.getElementById("sphereT").style.visibility="visible";
document.getElementById("cylT").style.visibility="hidden";
}
else if (shape=="sphere")
{
shape = "cylinder";
document.getElementById("sphereButton").style.visibility="collapse";
document.getElementById("cylButton").style.visibility="visible";
document.getElementById("cylVonMises").style.visibility="visible";
document.getElementById("axialStress").style.visibility="visible";
//drawing
document.getElementById("sphereT").style.visibility="hidden";
document.getElementById("cylT").style.visibility="visible";
}
else
{
//nothing
}
}
function compute(f) {
//var currentTime = new Date();
var or = parseFloat(f.outr.value);
var ir = parseFloat(f.innr.value);
var op = parseFloat(f.outp.value);
var ip = parseFloat(f.inp.value);
var radius = parseFloat(f.r.value);
var p1 = parseFloat(f.p.selectedIndex) + 1.;
var convLength = [0.001, 0.01, 1, 0.0254, (12*0.0254)];
var convPressure = [1000000, 1000, 1, 1/0.000145037738007,
1000*(1/0.000145037738007), 1000000*(1/0.000145037738007) ];
var convAnswer = [1/1000000, 1/1000, 1, 0.000145037738007,
0.000000145037738007, 0.000000000145037738007 ];
var cr = convLength[parseFloat(f.ropt.selectedIndex) ] ;
var cir = convLength[parseFloat(f.innro.selectedIndex)];
var cor = convLength[parseFloat(f.outro.selectedIndex)];
console.log(cor);
var cip = convPressure[parseFloat(f.innpo.selectedIndex)];
var cop = convPressure[parseFloat(f.outpo.selectedIndex)];
var cuanswer = convAnswer[parseFloat(f.answerUnits.selectedIndex)];
//converts values to meter and Pa before doing calculations
//convAnswer converts from meter, Pa to desired pressure units after
calculations
console.log(or);
console.log(ir);
console.log(op);
console.log(ip);
console.log(radius);
var or = cor*or;
var ir = cir*ir;
var op = cop*op;
var ip = cip*ip;
var radius = cr*radius;
console.log(or);
console.log(ir);
console.log(op);
console.log(ip);
console.log(radius);
if(ir > or)
{
alert("Error - Inner radius cannot be larger than outer radius");
return "Error"
}
if(or <=0 || ir <=0)
{
alert("Error - Radius must be a positive value");
return "Error"
}
if(radius > or || radius < ir)
{
alert("Error - Radius entered is not between inner and outer radius")
return "Error"
}
/*
//comment out after testing
f.r.value = radius.toPrecision(p1);
f.outr.value= or.toPrecision(p1);
f.innr.value= ir.toPrecision(p1);
f.outp.value = op.toPrecision(p1);
f.inp.value = ip.toPrecision(p1);
//
*/
var astress, cstress, rstress, vonmises;
f.astress.value= astress = (cuanswer*((ip*Math.pow(ir,2) -
op*Math.pow(or,2) )/(Math.pow(or,2) - Math.pow(ir,2)) ) ).toPrecision(p1) ;
f.cstress.value = cstress = (cuanswer*(((ip*Math.pow(ir,2) -
op*Math.pow(or,2) ) / (Math.pow(or,2) - Math.pow(ir,2))) -
((Math.pow(ir,2)*Math.pow(or,2) * (op - ip) ) / (Math.pow(radius,2) *
(Math.pow(or,2) - Math.pow(ir,2)))) ) ).toPrecision(p1);
f.rstress.value = rstress = (cuanswer*(((ip*Math.pow(ir,2) -
op*Math.pow(or,2) ) / (Math.pow(or,2) - Math.pow(ir,2))) +
((Math.pow(ir,2)*Math.pow(or,2) * (op - ip) ) / (Math.pow(radius,2) *
(Math.pow(or,2) - Math.pow(ir,2)))) ) ).toPrecision(p1);
if( isNaN(astress) || isNaN(cstress) || isNaN(rstress) ) {
//check
f.astress.value = "Error";
f.cstress.value = "Error";
f.rstress.value = "Error";
f.vstress.value = "Error";
return Error;
}
```

```
var cylstress = [astress, cstress, rstress];
var pcs = cylstress.sort(); //principal Cyl. stresses
console.log(pcs);
f.vstress.value = vonmises = ( Math.sqrt((Math.pow((pcs[0] - pcs[1]),2) +
Math.pow((pcs[1]-pcs[2]),2) +
Math.pow((pcs[2]-pcs[0]),2))/2) ).toPrecision(p1);
}
function computeSphere(f) {
//var currentTime = new Date();
var or = parseFloat(f.outr.value);
var ir = parseFloat(f.innr.value);
var op = parseFloat(f.outp.value);
var ip = parseFloat(f.inp.value);
var radius = parseFloat(f.r.value);
var p1 = parseFloat(f.p.selectedIndex) + 1.;
var convLength = [0.001, 0.01, 1, 0.0254, (12*0.0254)];
var convPressure = [1000000, 1000, 1, 1/0.000145037738007,
1000*(1/0.000145037738007), 1000000*(1/0.000145037738007) ];
var convAnswer = [1/1000000, 1/1000, 1, 0.000145037738007,
0.000000145037738007, 0.000000000145037738007 ];
var cr = convLength[parseFloat(f.ropt.selectedIndex) ] ;
var cir = convLength[parseFloat(f.innro.selectedIndex)];
var cor = convLength[parseFloat(f.outro.selectedIndex)];
console.log(cor);
var cip = convPressure[parseFloat(f.innpo.selectedIndex)];
var cop = convPressure[parseFloat(f.outpo.selectedIndex)];
var cuanswer = convAnswer[parseFloat(f.answerUnits.selectedIndex)];
//converts values to meter and Pa before doing calculations
//convAnswer converts from meter, Pa to desired pressure units after
calculations
console.log(or);
console.log(ir);
console.log(op);
console.log(ip);
console.log(radius);
var or = cor*or;
var ir = cir*ir;
var op = cop*op;
var ip = cip*ip;
var radius = cr*radius;
console.log(or);
console.log(ir);
console.log(op);
console.log(ip);
console.log(radius);
```

```
if(ir > or)
{
alert("Error - Inner radius cannot be larger than outer radius");
return "Error"
}
if(or <=0 || ir <=0)
{
alert("Error - Radius must be a positive value");
return "Error"
}
if(radius > or || radius < ir)
{
alert("Error - Radius entered is not between inner and outer radius")
return "Error"
}
/*
//comment out after testing
f.r.value = radius.toPrecision(p1);
f.outr.value= or.toPrecision(p1);
f.innr.value= ir.toPrecision(p1);
f.outp.value = op.toPrecision(p1);
f.inp.value = ip.toPrecision(p1);
//
*/
var astress, cstress, rstress, vonmises;
f.astress.value= astress = "-----" ;
f.rstress.value = rstress = (cuanswer*( ((op*Math.pow(or,3)) /
Math.pow(radius,3)) * ((Math.pow(radius,3)-Math.pow(ir,3))/(Math.pow(ir,3)-
Math.pow(or,3))) + ((ip*Math.pow(ir,3))/(Math.pow(radius,3))) *
((Math.pow(or,3)-Math.pow(radius,3))/(Math.pow(ir,3)-
Math.pow(or,3)))) ).toPrecision(p1);
f.cstress.value = cstress = (cuanswer*( ((op*Math.pow(or,3)) /
(2*Math.pow(radius,3))) *
((2*Math.pow(radius,3)+Math.pow(ir,3))/(Math.pow(ir,3)-Math.pow(or,3))) -
((ip*Math.pow(ir,3))/(2*Math.pow(radius,3))) *
((2*Math.pow(radius,3)+Math.pow(or,3))/(Math.pow(ir,3)-
Math.pow(or,3))) ) ).toPrecision(p1);
if( isNaN(cstress) || isNaN(rstress) ) {
//check
f.astress.value = "Error";
f.cstress.value = "Error";
f.rstress.value = "Error";
f.vstress.value = "Error";
return Error;
}
var cylstress = [astress, cstress, rstress];
var pcs = cylstress.sort(); //principal Cyl. stresses
console.log(pcs);
f.vstress.value = vonmises = "-----";
}
</SCRIPT>
</html>
```