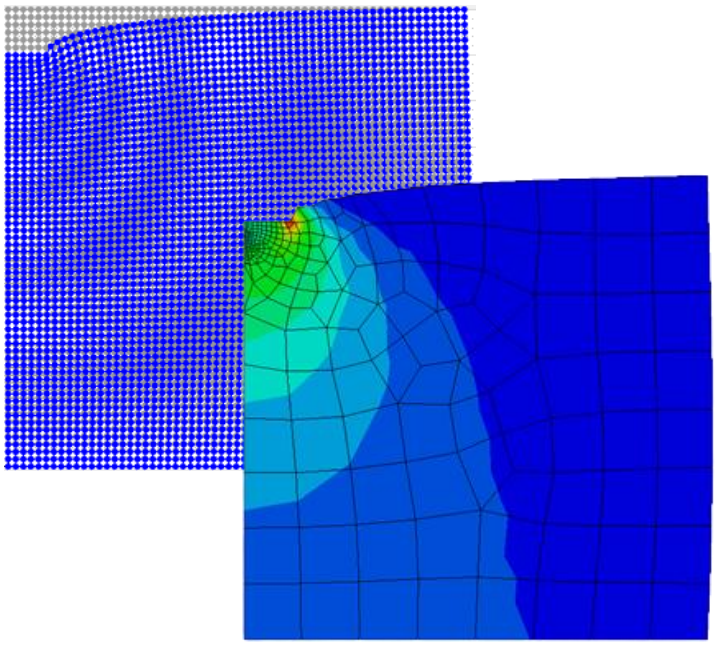
Linear Elastic *Planar* Indentation

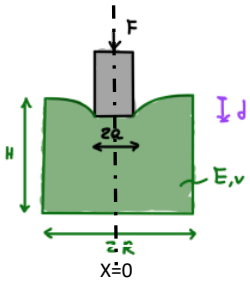
(MECH 462)

Allan Go

70263074



Introduction:



This project analyzes a planar rigid flat-punch indentation, assuming the prescribed displacement, d, of a linearly elastic specimen with modulus, E, and Poisson ratio, v. Assuming 2D plane strain, we can analyze a long rectangular specimen with a rigid indentation while reducing computational intensity. The specimen's height, H, and width, R, are much larger than the radius, *a,* of the punch, so we can assume an elastic half-space which can be compared with literature. MATLAB and Abaqus analyses can compare displacements and stresses with Johnson (1985). These require a linearly elastic structure to be valid. The goal of this project is to understand and apply finite element analysis with appropriate validation of results compared to theory.

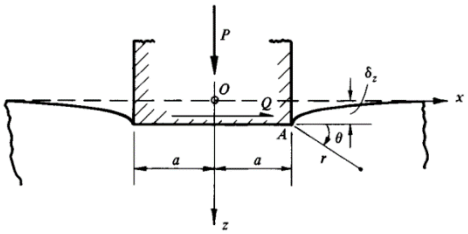
Methods:

A MATLAB model was first developed starting with a square mesh of linear elements. Boundary conditions along the bottom edge, as well as symmetry boundary conditions on the left edge, were added. Symmetry was used since more nodes and finer results could be achieved. Results will be the same on the other half of the structure and this allows for MATLAB and Abaqus' computational limits to be concentrated without duplicating data. Input parameters can be found in Table 1 in the appendix with exact specifications. A prescribed displacement, d, is modelled as the rigid punch on the top left of the structure with length, *a*. The mesh was modelled using varying dimensions to provide more accurate data (and more nodes) along the punch indentation. The model uses an elastic modulus, E = 1GPa, and Poisson ratio, ν = 0.3. To assume a rigid punch indentation, the elastic modulus of the structure must be much less than the modulus of the punch (Johnson, 1985). Therefore, 1GPa for a rubber or polymer structure was chosen while the punch is aluminum with E = 69Gpa. Displacements, stresses, and tractions were then plotted.

An Abaqus model was then developed with the same dimensions and properties as MATLAB to ensure consistency. Additionally, a partition was created along the top edge, at a distance 0.1m. This allows for precise mesh generation and can be used to match the number of nodes displaced by the punch in MATLAB. Linear interpolated elements with a quad-dominated mesh are being used. The shape of the mesh can be seen in the plots in the appendix. The displacements (U2), stresses (S11, S22), and tractions (RT1, RT2) are then generated in Abaqus and plotted. Data was exported into excel, and then MATLAB, for initial results validation.

Finally, results will be compared with the analytical vertical and horizontal traction solutions found in Johnson (1985) for a rigid flat-punch indentation with no-slip. Since the literature does not provide analytical solutions for the stress or displacements in the no-slip case, the tractions will be used for comparison instead. We will additionally compare our results to closed form solutions for the frictionless case of line loading of an elastic half space. Since this follows a frictionless assumption, we expect some deviation from our no-slip analysis. The analytical equations for frictionless displacement, stresses, and tractions are given as follows:

Displacement: $u_z = \delta_z - \frac{2(1-\nu^2)P}{\pi E}\ln\left(\frac{x}{a} + \left(\frac{x^2}{a^2} - 1\right)^{\frac{1}{2}}\right)$



Stress: $\sigma_1 + \sigma_1 \approx -\frac{2P}{\pi(2ar)^{\frac{1}{2}}}\sin(\theta/2)$

Tractions: $p(x) + iq(x) = \frac{2(1-\nu)}{(3-4\nu)^{\frac{1}{2}}}\frac{P+iQ}{\pi(a^2-x^2)^{\frac{1}{2}}}\left(\frac{a+x}{a-x}\right)^{i\eta}$, where $\eta = \frac{1}{2\pi}\ln(3-4\nu)$

Results:

Maximum Errors (MATLAB vs Abaqus)		Maximum Errors (MATLAB/Abaqus vs Literature)	
Displacement:	~4%	Displacement:	~14%
Stress:	~9%	Stress:	~6%
Tractions:	~6%	Tractions:	~7%

Maximum errors were calculated excluding the 2 elements nearest the edge of the punch. Figures 1 and 2 show the displacements and stress comparisons for MATLAB, Abaqus, and literature. Stresses plotted are the sum of σ_{11} and σ_{22} along the indentation to match the equation given in literature. The raw displacement plots can be found in Figures A1 and A2 in the appendix. Raw stress plots can be found in Figures A3 to A6. The MATLAB stresses are plotted at the coordinates of the midpoint of each element since the stress is for each element rather than each node like the displacements.

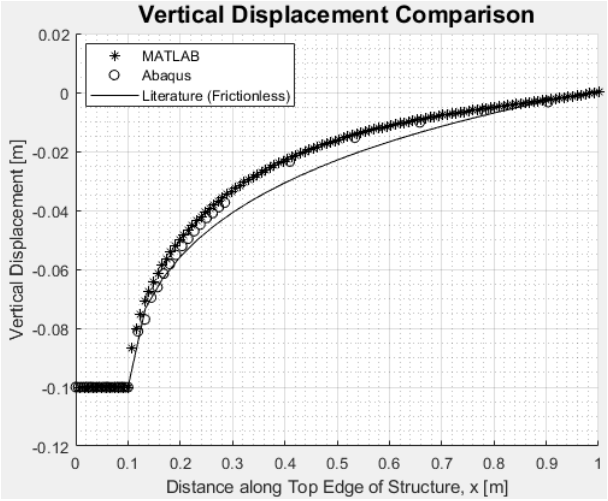


Figure 1: Comparison of Displacements

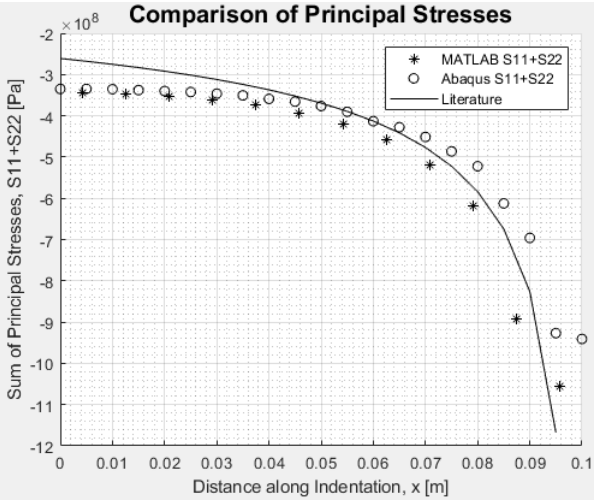


Figure 2: Comparison of Stresses

Figure 3 below shows the comparison of the vertical (p(x)) and horizontal (q(x)) traction forces in MATLAB, Abaqus, and literature. The MATLAB values were obtained from the vector of forces on the prescribed degrees of freedom. Abaqus data was plotted for the prescribed nodes' reaction force components (RT1 and RT2). For q(x), MATLAB and Abaqus values were scaled by 5/2 to match the plot given in literature.

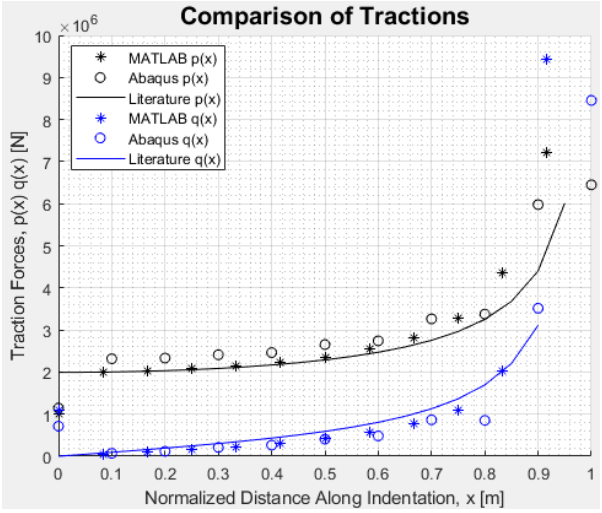


Figure 3: Comparison of Vertical p(x) and Horizontal q(x) Tractions

Validation and Limitations

While there are some large maximum error values in my analysis, these typically occur near the stress concentration created by the rigid punch. Thus, the maximum errors I presented exclude the two elements closest to that point where the theoretical stress would be infinite. Overall, my MATLAB and Abaqus results seem to validate each other particularly with respect to displacements and tractions. The shapes and magnitudes of my plots seem to match within an explainable margin and are very similar except near the stress concentration. Discrepancies in the displacement compared to literature is likely due to the no-slip condition in my analysis versus the frictionless case in literature. This is also the only scenario where an error of 10% or greater is seen. The stresses in my analysis show some discrepancies for data close to the edge of the punch indentation at x = 0.1. This is due to computational limits as the stress concentration approaches a theoretically infinite value. In MATLAB this affects the element closest to the edge of the indentation since one of its nodes would be at an infinite stress concentration causing errors in analysis. Similarly, in Abaqus the mesh can only be refined within hardware limitations and elements would need to by infinitesimally small near the stress concentration to avoid any error. At the centre of the punch at x = 0, my analysis shows unexpected values for the tractions which may be a result of the way MATLAB determines forces at a node relative to its neighbours. It could also indicate improper application of symmetry boundary conditions, but we would likely see discrepancies in the displacement and stress analyses which is not the case.

Comparing to literature there are some inherent errors with computational limits. I have tried to mitigate these as much as possible by increasing the number of nodes in MATLAB to the maximum possible, refining my Abaqus mesh near the area of interest, and utilizing symmetry. If unlimited computational power were available, error compared to the closed form solution could be minimized. Additionally, even though my Abaqus model could be refined further with quadratic interpolated elements and additional features, there is a need to maintain consistency between MATLAB and Abaqus. Ideally 10% error or less is optimal and my analyses meet this margin for most of the analyses conducted. This suggest the results I obtained are valid and there were no major inexplicable features.

Additional limitations and room for improvement include refined meshes in MATLAB and Abaqus, or testing different input parameters. The MATLAB analysis could benefit from concentrating more nodes near the rigid indentation like Abaqus. This could target the analysis more optimally but would be more time consuming to implement. Different input parameters could also be tested such as different elastic moduli or larger dimensions compared to the punch size. This would improve the analysis by ensuring our elastic half space assumption is exceeded. Most of the limitations arise from computational factors. We imposed certain assumptions to remedy this, but these also limit our analysis compared to the real scenarios that they approximate. This is often the case for computer models but demonstrates that we can still accurately validate and come to conclusions about our models using our understanding of theory. Accounting for discrepancies and using our understanding of the problem can allow us to interpret results and use our judgement to make appropriate decisions based on them.

References:

K. Johnson (1985). Contact Mechanics. Cambridge: Cambridge University Press.
doi:10.1017/CBO9781139171731

Appendix A – Additional Figures and Data:

Shared Parameters	*MATLAB Parameters*	*Abaqus Parameters*
• *$E = 1x10^9$ Pa* • *$v = 0.3$* • *$R = H = 1$ m* • *$a = 0.1$ m* • *$d = -0.1$ m* • *$E_{punch} = 69x10^9$ Pa*	• *60x60 Linear Elements (For Contour Plots)* • *120x120 Linear Elements (For Displacement, Stress, and Traction Data – the maximum capable for my computer's memory)*	• *Quad-Dominated Element Shapes, Linear Interpolated Standard Model* • *10 or 12 Nodes along punch indentation to match MATLAB*

Table 1: Summary of Input Parameters

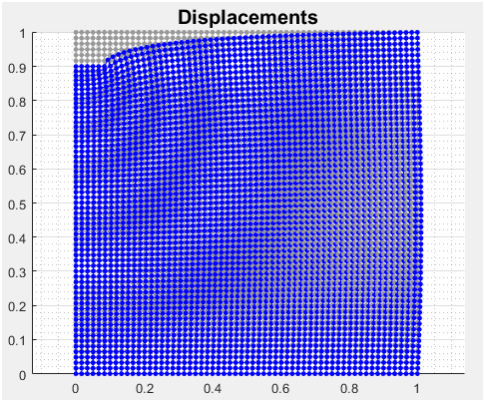


Figure A1: MATLAB nodal displacements

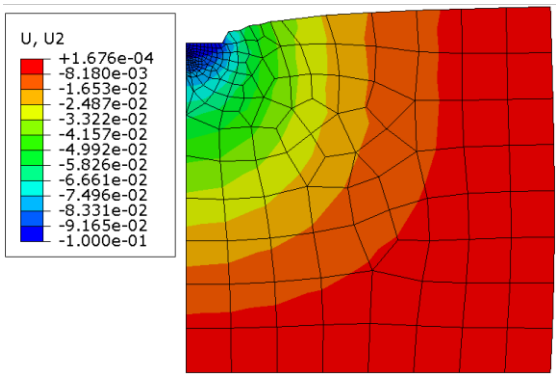


Figure A2: Abaqus Vertical Displacements

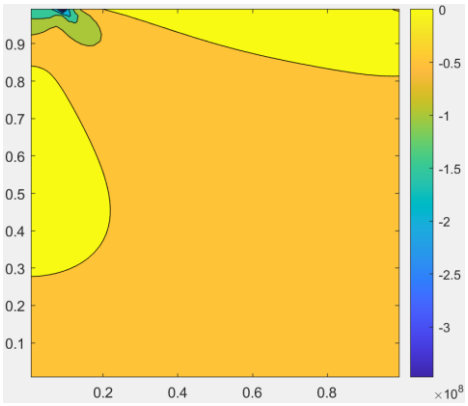


Figure A3: MATLAB Principal Stresses in x

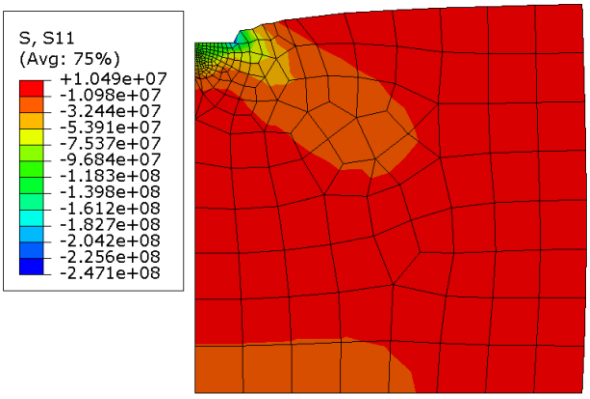


Figure A4: Abaqus Principal Stresses in x

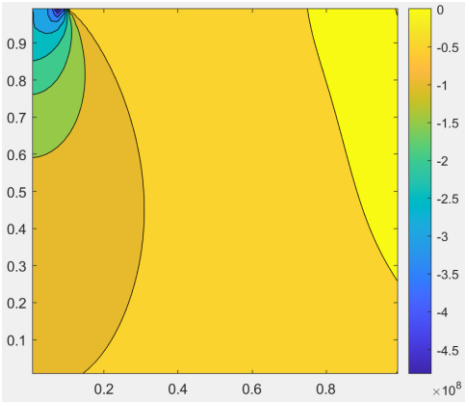


Figure A5: MATLAB Principal Stresses in y

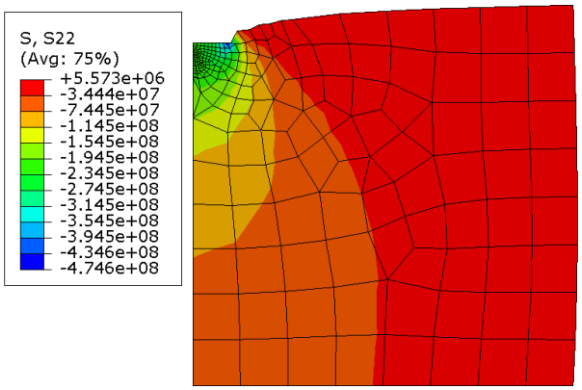


Figure A6: Abaqus Principal Stresses in y

Appendix B – MATLAB Code:

```
%%% a 2D continuum structure with Prescribed Displacement%%%
clear;
format long g

%%%%% VARIABLES %%%%%
% Plot parameters
nms = 3; % node marker size for displacements plot

% Material properties
E = 1*10^9; % Young modulus in [N/m^2]
v = 0.3;

% Size of the structure
R = 1; % R, the length of the structure in x-direction [m]
H = 1; % H, the length of the structure in y-direction [m]

% Size of the punch
a = 0.1*R; % radius, a, of the punch

% Applied external Force of the punch P (for literature equations)
P = 0.58e8; % [N]

% The size of the prescribed vertical displacement
delta = -0.1; % [m] downwards

% Mesh
nx = 120; % n. elements in x (nx = 100 nodes corresponds to 10 on the punch which is
the same we defined in abaqus)
ny = 120;
m = nx*ny; % total n. of elements
Dx = R/nx; % size of the element along x
Dy = H/ny;


%%%%% SETUP %%%%%
% Nodes
N = (nx+1)*(ny+1); % total n. of nodes

% Node grid - we use here a grid numeration
ix = [1:nx+1];
iy = [1:ny+1];

% Node coordinates
x = (ix-1)*Dx;
y = (iy-1)*Dy;
[xn,yn] = meshgrid(x,y);

% Stiffness matrix
K = zeros(2*N,2*N);
Em = [(1-v) v 0;
 v (1-v) 0;
 0 0 (1-2*v)/2]*E/((1+v)*(1-2*v)); % plane strain
```

```
T = [1,0,0,0;
 0,0,0,1;
 0,1,1,0];

% Gauss points
g(1,:) = 1/sqrt(3)*[-1,-1]; % the matrix g contains the coordinates of the Gaussian
points
g(2,:) = 1/sqrt(3)*[1,-1];
g(3,:) = 1/sqrt(3)*[-1,1];
g(4,:) = 1/sqrt(3)*[1,1];

for ex = 1:nx
 for ey = 1:ny
 i = (ey-1)*(nx+1)+ex;
 j = i+nx+1;
 x1 = xn(ey,ex); x2 = xn(ey,ex+1); x3 = x1; x4 = x2;
 y1 = yn(ey,ex); y2 = y1; y3 = yn(ey+1,ex); y4 = y3;
 Kq = 0;

 for gg = 1:4 % the four Gauss points
 xx = g(gg,1);
 yy = g(gg,2);
 Nxx = [-0.25*(1-yy), 0.25*(1-yy), -0.25*(1+yy), 0.25*(1+yy)];
 Nyy = [-0.25*(1-xx), -0.25*(1+xx), 0.25*(1-xx), 0.25*(1+xx)];
 J = [Nxx*[x1;x2;x3;x4], Nxx*[y1;y2;y3;y4]; Nyy*[x1;x2;x3;x4], Nyy*[y1;y2;y3;y4]];
 dJ = det(J);
 iJ = inv(J);
 A = T*[iJ,zeros(2,2);zeros(2,2),iJ];
 G = [Nxx(1) 0 Nxx(2) 0 Nxx(3) 0 Nxx(4) 0;
 Nyy(1) 0 Nyy(2) 0 Nyy(3) 0 Nyy(4) 0;
 0 Nxx(1) 0 Nxx(2) 0 Nxx(3) 0 Nxx(4);
 0 Nyy(1) 0 Nyy(2) 0 Nyy(3) 0 Nyy(4)];
 B = A*G;
 Kq = Kq + B'*Em*B*dJ;

 end
 edofs = [2*i-1,2*i,2*(i+1)-1,2*(i+1),2*j-1,2*j,2*(j+1)-1,2*(j+1)];
 K(edofs,edofs) = K(edofs,edofs)+Kq;
 end
end

u = zeros(2*N,1); % the function 'zeros' gives a matrix by defauls, if one of the
sizes is 1, that gives a tensor

% Boundary conditions
ixfix = [1:nx+1]; % position of the fixed nodes in the node grid
iyfix = 1;
ifix = (iyfix-1)*(ny+1)+ixfix;

% Additional symmetry boundary conditions
% the DOFs in the x direction normal to the plane of symmetry
sym_dofsx = 1;
sym_dofsy = [2:ny+1];
sym_dofs = (sym_dofsy-1)*(ny+1)+sym_dofsx;
```

```
fix_dofs = [2*ifix-1 2*ifix 2*sym_dofs-1];

% Loads
f = zeros(2*N,1);

% Prescribed displacements
ixpr = [1:round(a/Dx)]; % round the number of prescribed nodes to the nearest node -
assuming we have at least one node prescribed
iypr = (ny+1);
ipr = (iypr-1)*(ny+1)+ixpr;
pr_dofs = [2*ipr-1 2*ipr];
u(2*ipr) = delta;
u(2*ipr-1) = 0;

% Solution
free_dofs = setdiff([1:2*N],fix_dofs);
freeNP_dofs = setdiff(free_dofs,pr_dofs); % free dofs with Non Prescribed
displacements
K_ff = K(freeNP_dofs,freeNP_dofs);
K_fp = K(freeNP_dofs,pr_dofs);
K_pf = K(pr_dofs,freeNP_dofs);

K_pp = K(pr_dofs,pr_dofs);

u(freeNP_dofs) = K_ff\(f(freeNP_dofs)-K_fp*u(pr_dofs));

% Prescribed DOFs
f(pr_dofs) = K_pf*u(freeNP_dofs)+K_pp*u(pr_dofs);
F_out = f(pr_dofs);

%%%%% ANALYSIS %%%%%
% Displacement Plotting Nodes
disp_dofs = [2*(nx+1)*(ny+1)-2*(nx+1)+1:2*(nx+1)*(ny+1)];
punch_displacements = u(disp_dofs);

% MATLAB numbers for comparison with literature
x_compare = (0/a):(Dx/a):((a-Dx)/a);
q_x = F_out(1:length(F_out)/2)*(5/2); % scaled by 5/2 since literature scales by 5/2
on their plots on pg 37
p_x = -1*F_out(length(F_out)/2+1:length(F_out));

% LITERATURE plotting the equation 2.69 given in the literature
% Comparing Tractions for no-slip case
x_lit_p = 0:0.005:0.095;
x_lit_q = 0:0.005:0.09;
x_normed_p = x_lit_p./a;
x_normed_q = x_lit_q./a;
F1 = P; % the literature defines positive punch Force (P in literature) as downwards
(+)
Q = 0;
eta = (1/2/pi)*log(3-4*v);
lit_p = (2*(1-v)/(3-4*v)^(1/2))*(F1+1i*Q)./(pi*(a^2-
x_lit_p.^2).^(1/2)).*((a+x_lit_p)./(a-x_lit_p)).^(1i*eta);
lit_q = (2*(1-v)/(3-4*v)^(1/2))*(F1+1i*Q)./(pi*(a^2-
x_lit_q.^2).^(1/2)).*((a+x_lit_q)./(a-x_lit_q)).^(1i*eta);
```

```
% Comparing equation 2.66 for vertical displacement from literature in the
% frictionless case
x_uz_lit = [0:R/30:R];
uz_lit = delta + (2*(1-v^2)*P)/(pi*E)*(log(x_uz_lit/a+((x_uz_lit.^2)/(a^2)-
1).^(1/2))); % pg 38 eqn 2.66

% Comparing equation 2.65a for sum of principal stresses
theta = 180; % degrees
r = [a:-a/20:0];
sigma_sum = -2*P*sind(theta/2)./(pi*(2*a*r).^(1/2));

% Checking average contact stress sigmaC/E<=0.1 for linear elastic
% assumption:
sigmaC = trapz(r(1:end-1), sigma_sum(1:end-1)); % aproximate since sigma->inf at 'a'
LinElastCheck = sigmaC/a/E;

% ABAQUS Data
% Abaqus tractions with 10 nodes on the punch (this matches nx = 100 in matlab)
Aba_x = [0 0.00999999 0.020000011 0.030000001 0.039999992 0.050000012 0.060000002
0.069999993 0.080000013 0.090000004 0.099999994];
Aba_p = -1*[-1145279.125 -2316072.75 -2332994.25 -2411538.75 -2462961.25 -2653358.75
-2741150 -3262788 -3376709.5 -5978484.5 -6448409];
Aba_q = [285667.5 27391.84766 46862.62109 83357.63281 105207.2109 162079.3906
192889.2656 345686.5 340590.6875 1406705.375 3381705.5].*(5/2); % scaled by 5/2 to
match literature

% Abaqus Displacements
Aba_disp_coords = [0 0.004999995 0.00999999 0.014999986 0.020000011 0.025000006
0.030000001 0.034999996 0.039999992 0.044999987 0.050000012 0.055000007 0.060000002
0.064999998 0.069999993 0.074999988 0.080000013 0.085000008 0.090000004 0.094999999
0.099999994 0.11909838 0.132430792 0.14445135 0.156503811 0.168147013 0.180046082
0.191537693 0.20328775 0.214898482 0.226575002 0.238074496 0.250272751 0.26183933
0.273643523 0.285146147 0.409303457 0.533932745 0.657952428 0.781127036 0.9033916];
Aba_disp_y = [-0.100000001 -0.100000001 -0.100000001 -0.100000001 -0.100000001 -
0.100000001 -0.100000001 -0.100000001 -0.100000001 -0.100000001 -0.100000001 -
0.100000001 -0.100000001 -0.100000001 -0.100000001 -0.100000001 -0.100000001 -
0.100000001 -0.100000001 -0.100000001 -0.100000001 -0.081078283 -0.077019833 -
0.069590062 -0.066093795 -0.061480757 -0.058281649 -0.055134017 -0.05213026 -
0.049633589 -0.04708001 -0.044878073 -0.04258687 -0.040950429 -0.03893711 -
0.037387468 -0.023433032 -0.015344316 -0.010175573 -0.006223801 -0.003271678];

% Abaqus stresses S11 S22 for E = 1e9
Aba_S_coords = [0 0.004999995 0.00999999 0.014999986 0.020000011 0.025000006
0.030000001 0.034999996 0.039999992 0.044999987 0.050000012 0.055000007 0.060000002
0.064999998 0.069999993 0.074999988 0.080000013 0.085000008 0.090000004 0.094999999
0.099999994];
Aba_S11 = [-103653224 -103081208 -102437440 -102808128 -103145832 -103528696 -
104816656 -106307504 -109977760 -111759592 -115855632 -121608320 -130856352 -
131844800 -140071392 -152768000 -163193232 -199221536 -218119712 -275648448 -
403756864];
Aba_S22 = [-231267312 -231658416 -233146416 -235001200 -236926464 -238895712 -
241380832 -244301792 -248924800 -253953088 -260527328 -268720896 -282249344 -
295725440 -311365568 -333495488 -359169824 -413150304 -477931776 -651133952 -
537190272];
```

```
%%%%% PLOTTING %%%%%
%{
% Verify q(x)/p(x) relationship in literature and correct equation
figure(100)
hold on; grid on; grid minor
title('q(x)/p(x)','fontsize',15)
%plot(x_compare,p_x_what,'rs','Markersize',6)
plot(x_normed_p,imag(lit_p)./real(lit_p))
%}

% compare tractions
figure(200)
hold on; grid on; grid minor
title('Literature Equation 2.69 Plot','fontsize',15)
plot(x_compare,p_x,'k*','Markersize',6)
plot(Aba_x/a,Aba_p,'ko')
plot(x_normed_p,real(lit_p), 'k-')
plot(x_compare,q_x,'b*','Markersize',6)
plot(Aba_x/a,Aba_q,'bo')
plot(x_normed_q,(5/2).*imag(lit_q), 'b-')
xlabel('Normalized Distance Along Indentation, x [m]')
ylabel('Traction Forces, p(x) q(x) [N]')
legend('MATLAB p(x)', 'Abaqus p(x)','Literature p(x)','MATLAB q(x)', 'Abaqus
q(x)','Literature q(x)')


% compare displacements
figure(300)
hold on; grid on; grid minor
title('Vertical Displacement Comparison','fontsize',15)
plot([1/length(punch_displacements(2:2:end)):1/length(punch_displacements(2:2:end)):1
],punch_displacements(2:2:end),'k*')
plot(Aba_disp_coords,Aba_disp_y,'ko')
plot(x_uz_lit,uz_lit,'k-')
xlabel('Distance along Top Edge of Structure, x [m]')
ylabel('Vertical Displacement [m]')
legend('MATLAB', 'Abaqus', 'Literature (Frictionless)')


% Plot results
figure(1)
hold on; grid on; grid minor
title('Nodes','fontsize',15)
plot(xn(iyfix,ixfix),yn(iyfix,ixfix),'rs','Markersize',6)
plot(xn(iypr,ixpr),yn(iypr,ixpr),'bs','Markersize',6)
plot(xn,yn,'ko','Markersize',3)
plot(xn(sym_dofsy,sym_dofsx),yn(sym_dofsy,sym_dofsx),'g>','Markersize',6)
legend('fixed nodes','prescribed displacements nodes','all nodes','symmetry
condition','Location','NorthEastOutside')
axis equal;


%%%
figure(2)
```

```
title('Displacements','fontsize',15)
hold on; grid on; grid minor
grey = 0.6*[1 1 1];
for ey = 1:ny
 for ex = 1:nx
 x1 = xn(ey,ex); x2 = xn(ey,ex+1); x3 = x1; x4 = x2;
 y1 = yn(ey,ex); y2 = y1; y3 = yn(ey+1,ex); y4 = y3;
 i = (ey-1)*(nx+1)+ex;
 j = i+nx+1;
 u1 = u(2*i-1,1); v1 = u(2*i,1);
 u2 = u(2*(i+1)-1,1); v2 = u(2*(i+1),1);
 u3 = u(2*j-1,1); v3 = u(2*j,1);
 u4 = u(2*(j+1)-1,1); v4 = u(2*(j+1),1);

 plot([x1,x2],[y1,y2],'k-o','Color',grey,'MarkerSize',nms,'markerfacecolor',grey)
 plot([x1,x3],[y1,y3],'k-o','Color',grey,'MarkerSize',nms,'markerfacecolor',grey)
 plot([x2,x4],[y2,y4],'k-o','Color',grey,'MarkerSize',nms,'markerfacecolor',grey)
 plot([x3,x4],[y3,y4],'k-o','Color',grey,'MarkerSize',nms,'markerfacecolor',grey)
 plot([x1+u1,x2+u2],[y1+v1,y2+v2],'b-o','MarkerSize',nms,'markerfacecolor','b')
 plot([x1+u1,x3+u3],[y1+v1,y3+v3],'b-o','MarkerSize',nms,'markerfacecolor','b')
 plot([x2+u2,x4+u4],[y2+v2,y4+v4],'b-o','MarkerSize',nms,'markerfacecolor','b')
 plot([x3+u3,x4+u4],[y3+v3,y4+v4],'b-o','MarkerSize',nms,'markerfacecolor','b')


 % plot strains and stresses
 ee11 = 0.5*((u2-u1)/(x2-x1)+(u4-u3)/(x4-x3));
 ee22 = 0.5*((v3-v1)/(y3-y1)+(v4-v2)/(y4-y2));
 ee12 = 0.5*((u3-u1)/(y3-y1)+(u4-u2)/(y4-y2)+(v2-v1)/(x2-x1)+(v4-v3)/(x4-x3));
 eev(ey,ex,:) = [ee11;ee22;ee12];
 sv(ey,ex,:) = Em*[ee11;ee22;ee12];
 Uv(ey,ex) = 0.5*[ee11,ee22,ee12]*Em*[ee11;ee22;ee12];
 xc(ey,ex) = 0.25*(x1+x2+x3+x4);
 yc(ey,ex) = 0.25*(y1+y2+y3+y4);
 end
end
axis equal

for splt = 1:3
 figure(10+splt)
 contourf(xc,yc,sv(:,:,splt))
 colorbar
 axis equal
end
for eplt = 1:3
 figure(20+eplt)
 contourf(xc,yc,eev(:,:,eplt))
 colorbar
 axis equal
end
figure(30)
contourf(xc,yc,Uv)
colorbar
axis equal
```

```
s11 = sv(:,:,1);
s11_top = s11(nx,1:nx*a);
s22 = sv(:,:,2);
s22_top = s22(nx,1:nx*a);
element_coord = 1/length(s11_top)*a;
stress_coords = [element_coord/2:element_coord:a];

% compare stresses
figure(400)
hold on; grid on; grid minor
title('Comparison of the Sum of Principle Stresses Along The
Indentation','fontsize',15)
plot(stress_coords,s11_top+s22_top,'k*')
plot(Aba_S_coords,Aba_S11+Aba_S22,'ko')
plot(r,flip(sigma_sum),'k-')
xlabel('Distance along Indentation, x [m]')
ylabel('Sum of Principal Stresses, S11+S22 [Pa]')
legend('MATLAB S11+S22', 'Abaqus S11+S22', 'Literature')
```