Portal Frame Study:

A comparison between traditional W-shapes and a PLATO optimized topology

Jose Vera
ME EN 6510
December 10, 2020

Contents

01	Introduction	1
02	Geometry	1
03	Material Model	2
04	Boundary Conditions	3
05	Mesh	4
06	Results	7
07	Verification and Validation	8
08	Discussion and Conclusions	9
09	References	10
10	Appendix	11

1 Introduction

In this report, the analysis of a steel portal frame is presented. Steel portal frames are structures, typically composed of w-shapes, employed in buildings where lateral forces need to be resisted while allowing space for occupants to pass through. This report exhibits how such a frame can be modeled in 3D and how it reacts to the applied loading. Both the stress and displacements are analyzed, and a topology analysis of the portal frame space is also presented. This report aims to show how optimized topologies can be used as alternative solutions to traditional portal frames.

2 Geometry

The steel portal frame has a beam length of 15 ft. and a column height of 10 ft. Both the beams and columns will consist of a W16X31 shape. To adequately resist the lateral loads, the beam must be able to completely transfer shear and moment to the columns; in practice, this is typically achieved through a moment connection such as a continuity plate [1]. The columns must similarly be able to transfer forces to the ground which, in practice, is typically achieved by welding the column to a base plate. For simplicity, this project only includes the welded column-baseplate connection. In ABAQUS, this complete transfer of forces was achieved by assembling instances of the different parts such that the beams sit on top of the columns and the columns on top of the baseplates. The assembly of part instances was merged to create a part that ensures the frame acts as one unit.

2.1 Model Simplifications and Assumptions

The model has several key simplifications that must be stated. The most important is the connection between the beam and the columns. As stated previously, this connection would typically be detailed as a moment connection by way of a continuity plate. To simplify the modeling, this connection was omitted and, instead, the beam was modeled to directly bear on the columns. Transfer of forces was ensured by merging the entire assembly to create one part. The next simplification is with regards to the region where the loads were applied. In a real-world application, the gravity load would be applied over the entire top flange of the beam and the lateral load over the entire column. To simplify the meshing in ABAQUS, the gravity load was only applied to the top surface of the beam web and the lateral load only on the side surface of the beam web. This ensures that no unrealistic distortion occurs throughout the frame.

2.2 3-D Geometry

The geometry of the portal frame used in the analysis is based on the typical portal frame used in commercial structural engineering. To create the 3D geometry, the sections for the beams, columns, and baseplates were drawn and then extruded to their appropriate dimension. The extrusion dimensions for the beam, columns, and baseplates were 180 in., 120 in., and 1 in., respectively. The frame geometry comes together by way of instance assembly as stated in the previous sections. Figure 1 shows the frame assembly.

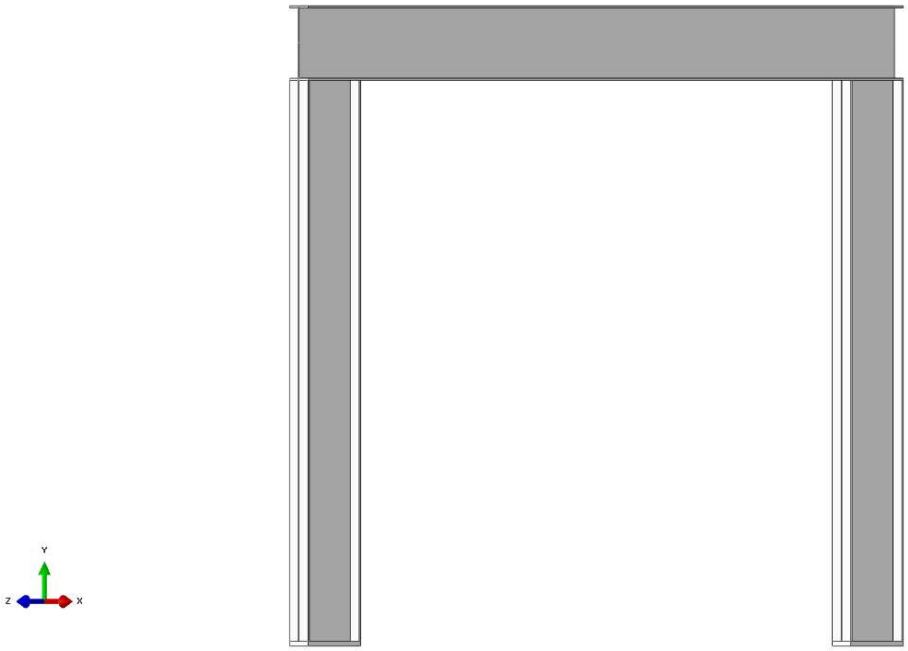

Figure 1: 3D Geometry of frame assembly.

3 Material Model

A5722 Gr. 60 steel represents the material used for all parts. This type of steel is the most widely available for w-shapes, the most used shape for moment frames in commercial construction; this makes it the most practical for this project. Yielding deformations in portal frames are considered a serviceability failure and, consequently, the steel was modeled as a linearly elastic isotropic material, both in the ABAQUS and PLATO model. Table 1 below shows the steel material parameters used for elastic and density properties.

Parameter	Value	Unit
Density	0.282	lb/in^3
Modulus of Elasticity	29,000,000	lb/in^2
Yield Stress	60,200	lb/in^2
Poisson's Ratio	0.303	

Table 1: A572 Gr. 60 Steel Parameters from MatWeb [2].

4 Boundary Conditions

There are both displacement and force boundary conditions imposed on the model.

4.1 Displacements

Portal frames are used to resist the horizontal shear at the top of the columns and therefore their supports should be able to resolve both shear and moments to the ground. In ABAQUS, this fixity was represented by fixing translations and rotations at the bottom of the column baseplates. In PLATO, the bottom surfaces of the columns in the bounding space representation of the frame were set to fixed in the input file.

4.2 Forces

The loading on the steel portal frame will consist of a 1000 lb/ft downward load on the top beam and a 10,000 lb horizontal load on the side of the beam, both modeled with applied pressures. These forces are based on the gravity and earthquake loading values for a typical two-story office building. In ABAQUS, the gravity and earthquake loads were applied at the top and side surfaces of the web partition, respectively; in GMSH, they were both applied on the top surface of the beam bounding box. The pressures were scaled to the applied areas in ABAQUS and GMSH such that their resultant forces equal the loads previously stated. See appendix C for the load calculations. An illustration of the applied loads and boundary conditions is illustrated in Figure 2 below.

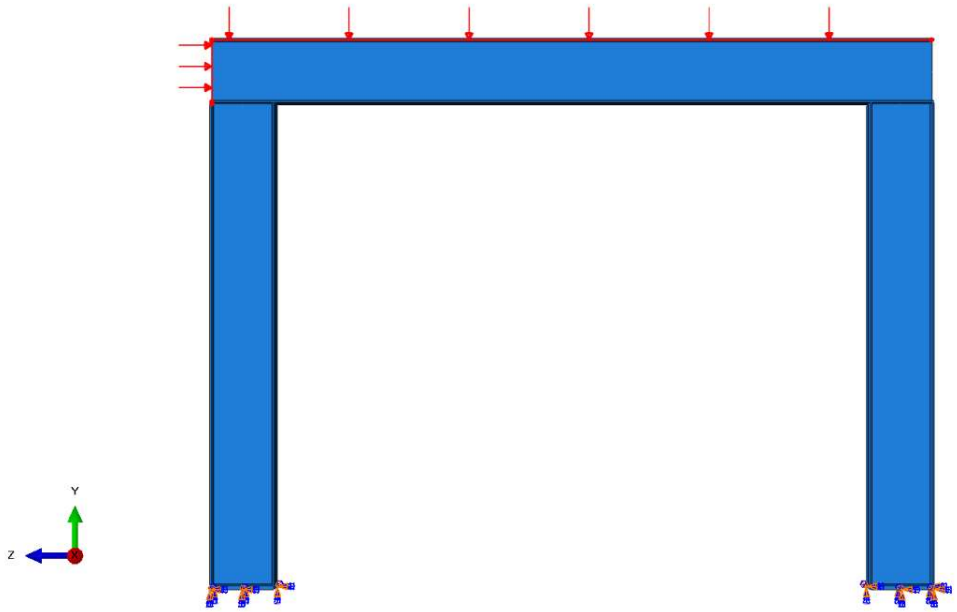

Figure 2: Portal frame illustration of applied loads and support conditions.

5 Mesh and setup

The parts used for my model are all 3D, deformable-body. All elements are 20-node quadratic bricks, a subset of the 3D stress family. The frame was assigned global seeds and the size was progressively decreased until the model results were independent of the mesh.

5.1 Setup: Python-ABAQUS Scripting

All the pre and post-processing associated with modeling this portal frame was done through a python code by way of the ABAQUS scripting interface [3]. This includes part creation, material creation, instance creation, frame assembly, frame meshing, BC definition, job submitting, iteration through different seed sizes, and result extraction. See Appendix D for the script.

5.2 Mesh Convergence

To ensure mesh elements of adequate dimensions, partitioning of the w-shapes was necessary. The shapes were partitioned at the web faces such that the web extends the full depth of the members and that the flanges on either side are separate. Figure 3 below shows this:

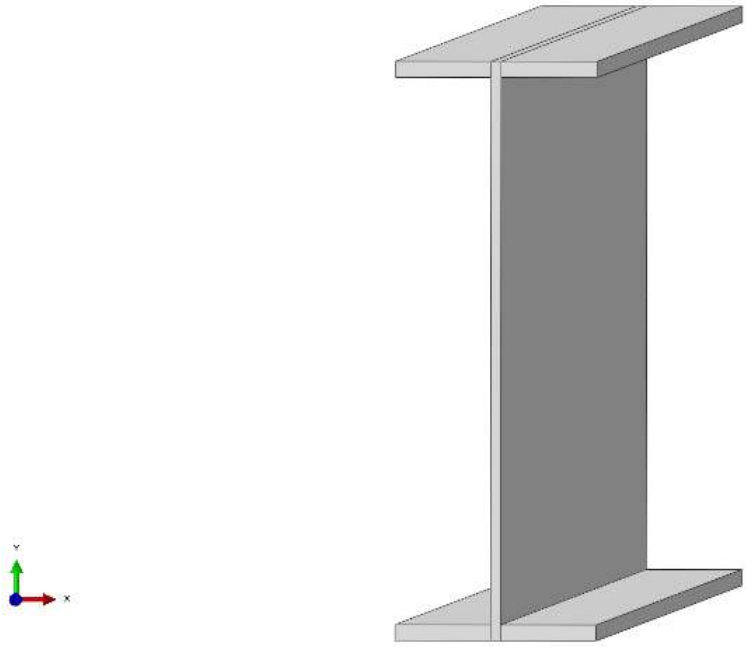

Figure 3: Partitioning of typical w-shape in the frame.

Using global part seeds as opposed to local edge seeds made the meshing very straight forward. ABAQUS does a good job at meshing using global part seeds when the geometry consists exclusively of rectangular elements. Furthermore, using global seeds simplified the python script. Figure 4 shows the converged frame mesh when global part seeds were set to a size of 0.75.

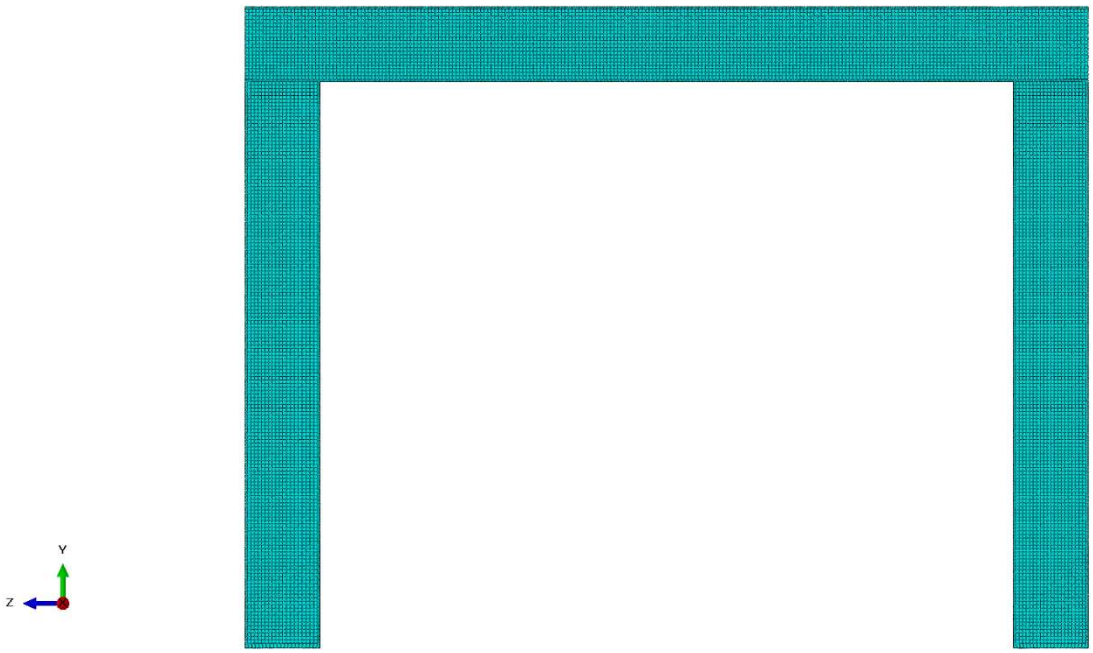

Figure 4: Frame mesh at global part seed size of 0.75.

The global part seed size was gradually decreased until the results were independent of the mesh and the node size restriction of the license was reached. This convergence study was conducted by the python script, which iterated through sizes of 48, 24, 12, 6, 3, 1.5, and 0.75. In each iteration, the script also prompted ABAQUS to report displacements and stress values for each node. The script also prompts the creation of displacement and stress plots for the deformed shape. Maximum vertical displacement and Von Mises stress versus the number of elements are plotted in Figure 5 below:

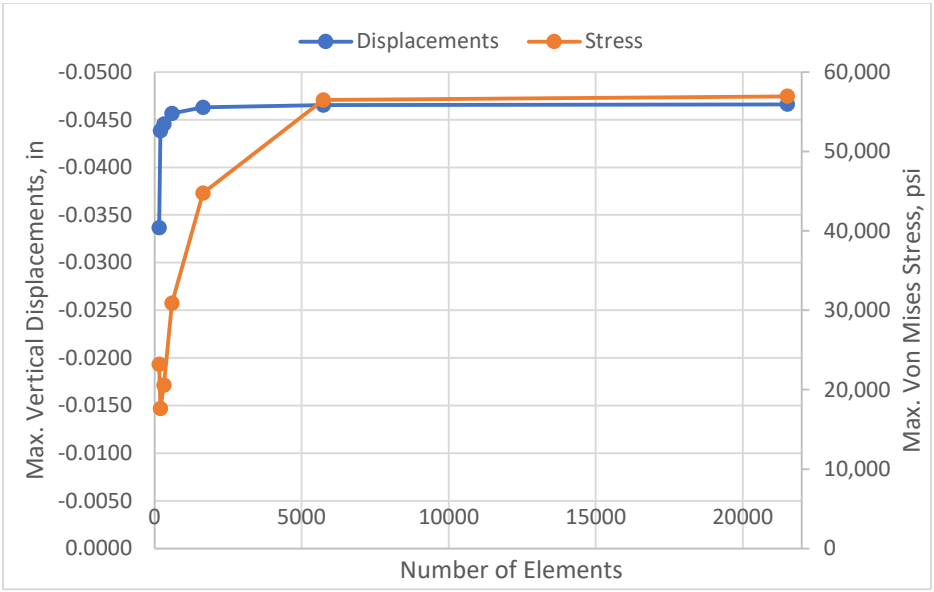

Figure 5: Convergence plot of maximum Von-Mises stress and vertical displacement.

The plot shows that, as the number of elements increases, the maximum vertical displacements and stress tend towards a converged value. More iterations would show how the values continue to converge, but the school license of ABAQUS cannot run jobs with the elements associated with the next global part seed size of 0.375. Appendix B contains tables with the values used to generate Figure 5.

5.3 Analysis Type

The type of ABAQUS analysis used was ABAQUS explicit.

5.4 Topology Optimization Mesh

There was also another model created specifically for topology optimization of the portal frame. Using a proprietary meshing software package from Sandia National Laboratory called Cubit, a bounding box was created that represents the space occupied by the frame; mesh size was set at 1 in. Figure 5 below shows a GMSH representation of the initial bounding box. This volume represents the volume bounded by the W16x31 frame and where PLATO is allowed to place and remove material to maximize stiffness and minimize stress localization.

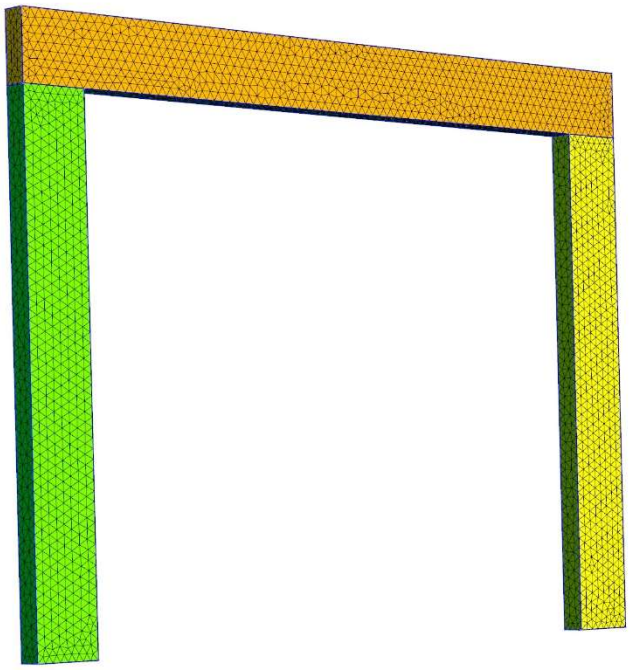

Figure 6: GMSH representation of bounding box used for topology optimization.

6 Results

There were a few main results that were extracted from ABAQUS. The first result was the maximum displacement in the y-direction (U2) corresponding to the gravity load on the top of the beam. The second was the maximum displacement in the z-direction (U3) corresponding to the lateral load on the side of the beam. The last was the maximum Von Mises stress in the frame. Table 2 below shows the results of the converged mesh.

Global Part Seed Size	0.75
Maximum Horizontal Displacement, in.	-0.1672
Maximum Vertical Displacement, in.	-0.0466
Maximum Von Mises Stress, psi.	56,955

Table 2: Results associated with the converged mesh seed size of 0.75.

Note that the maximum stress is below 60,000 psi and thus no steel yielding occurs. See Figure 6 below for an illustration of the maximum Von Mises stress. See Appendix AA for an illustration of the displacement.

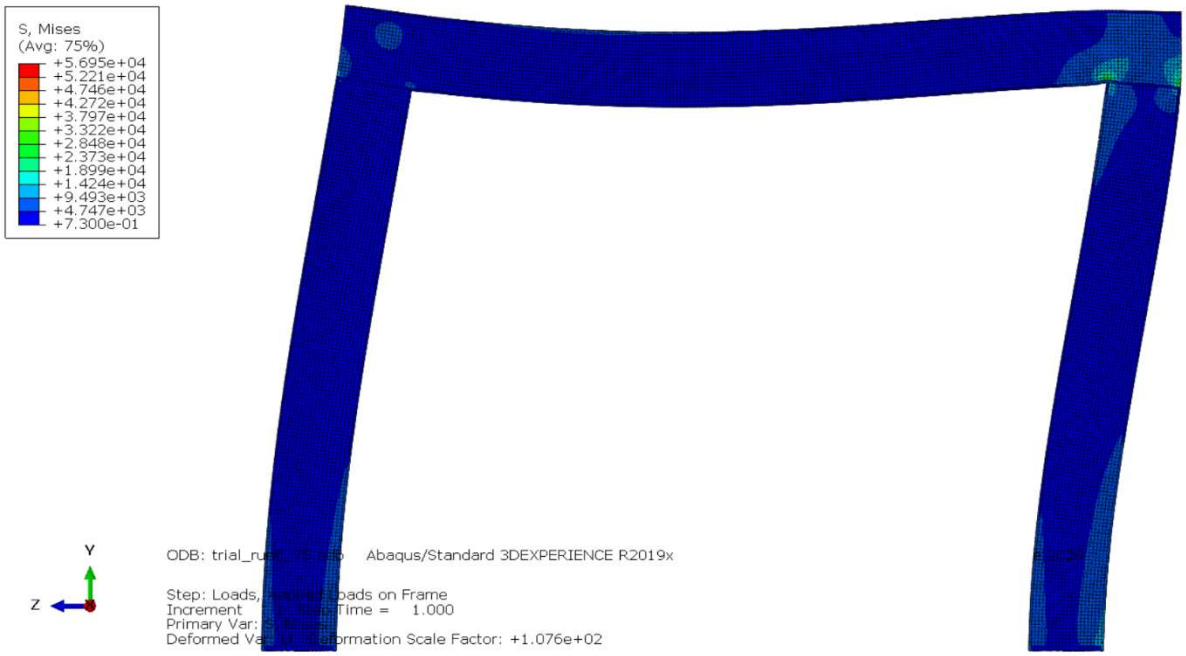

Figure 7: Illustration of Von Mises stress on the converged mesh. Legend values are in psi.

6.1 Topology Optimization

Dr. Aguilo from Sandia National Laboratories provided us with the Platform for Topology Optimization (PLATO) [4]. This tool was used to generate optimized designs based on a volume constraint with a maximize stiffness objective; the geometry was subject to the boundary

conditions mentioned previously. See Appendix E for the PLATO input code. Figure 7 shows the optimized topology for this portal frame problem.

Figure 8: Topology optimized portal frame. The colors represent contributions to stiffness and system stability, blue means little contribution and red means significant contribution.

It is interesting to see spaces in the optimized geometry where very little material is needed. Note that this result corresponds solely to lateral loading acting towards the right. For this case, material in the left column was almost completely eliminated. The beam and the right column were significantly hollowed out in the spots where material is not needed to efficiently carry the loads from their application point to the structural supports. Appendix AA shows a few more loading scenarios and their corresponding optimized topologies.

7 Verification, Validation & Uncertainty Quantification

7.1 Verification

Verification is defined as the process of comparing the FEA result to a known analytical solution. For this problem, a simple SAP2000 2D static model was created to model the portal frame with the applied loads and use as a pseudo analytical solution. SAP2000 is a software package typically used in structural engineering for analysis. Maximum horizontal and vertical displacements were extracted and used as the benchmark for the FEA results. Table 3 shows the results from that model.

	SAP2000 Result
Max. Horizontal Displacement at Beam, in	0.12382
Max. Vertical Displacement at Beam, in	0.04884
Max. Stress, psi	58,000

Table 3: Benchmark results from SAP2000

Overall, SAP2000 and ABAQUS yield very similar values. In particular, the maximum vertical displacement relative difference is less than 5%. Appendix B contains tables with the relative error calculation of the maximum horizontal and vertical displacement.

7.2 Validation

The validation process for the model is more complicated because of the lack of test data; portal frames are typically part of the lateral force-resisting system and there is no test available where the load is idealized as a point load at the top of the column with the cross-sections used. To properly validate my model, an experiment would be prepared where the beam bottom flanges are welded to the top of the columns, and the bottom of the columns welded to a base plate fixed to the ground. The gravity load would be applied solely on the web-projection region of the top flange of the beam to match the ABAQUS model. The lateral load would be applied directly on the side edge of the beam. Measurements of vertical and horizontal displacements, as well as maximum stress, would be made to compare to the FEA results presented herein.

7.3 Uncertainty Quantification

There are two main sources of uncertainty in my model: beam-to-column connections and areas of load application. It was previously stated as one of the main assumptions that the beam transfers all forces to the adjacent columns perfectly as if they are one part. This is not representative of real-world construction though, and the results of an FEA analysis that models the connection differently may give drastically different results, especially for stress. The areas of load application could potentially also change the results significantly. In real applications, portal frames are near the exterior and the slab weight produces a downward force over the entire top flange of the beam. In addition, it would apply a moment about the longitudinal axis of the beam which is not even considered in this report. The lateral load would not be applied solely on the side surface of the beam web and instead would be the result of the slab mass above accelerating as a result of an earthquake shaking the ground. Changing any of these two things would likely produce results that are different than those presented in this report.

8 Discussion and Conclusions

The frame model behaves in accordance with was is expected from a portal frame; both columns sway to the right due to the lateral load and the beam deflects downward due to the gravity load. The maximum stress throughout the frame remains below yield stress meaning that the structure does not suffer any permanent deformations. It is worth noting, however, that the maximum

stress in the structure occurs at the beam-column connection. In a real application, this connection needs to be designed knowing that it is the most likely failure location. An improvement on the FEA analysis of this report would be to try multiple connection geometries and measure their performance relative to each other. The initial design of such geometry should, to some degree, be inspired by the beam-column connection shown in the optimized topology.

Overall, it is demonstrated that a portal frame composed of W16x31 is more than enough to support the applied loading. Though the w-shape geometry used works, it does not mean it is the best solution we have. The topology optimization included in this report shows how little material needs to be used to adequately carry the applied loads. This geometry is interesting to look, but not practical to build in the present due to manufacturing concerns. Nevertheless, the advent of 3D printing and additive manufacturing are promising technologies that allow us to bring to life these geometries. The current cost of printing these geometries limits the technology to very high budget applications. As the technology scales, it will surely make its way into commercial structural engineering, and eventually into small residential projects. At some point, we may even be entering our homes through an entryway that is supported by a geometry similar to the optimized frame presented in this report.

9 References

[1] J. E. R. O. M. E. F. HAJJAR, R. O. B. E. R. T. J. DEXTER, S. A. R. A. D. OJARD, Y. A. N. Q. U. N. YE, and S. E. A. N. C. COTTON, "Continuity Plate Detailing for Steel Moment-Resisting Connections," 2003. [Online]. Available: https://www.aisc.org/globalassets/aisc/awards/tr-higgins/past-winners/continuity-plate-detailing-for-steel-moment-resisting-connections.pdf. [Accessed: 17-Nov-2020].

[2] MATWEB, *ASTM A572 Steel, grade 60*. [Online]. Available: http://matweb.com/search/datasheet.aspx?matguid=b96550fcbd964d6bb51a59dec4f0878f. [Accessed: 11-Nov-2020].

[3] MIT, "Using Python and the Abaqus Scripting Interface," 2017. [Online]. Available: https://abaqus-docs.mit.edu/2017/English/SIMACAECMDRefMap/simacmd-m-IntPythonandacl-sb.htm. [Accessed: 20-Nov-2020].

[4] "Plato Home," *Platform for Topology Optimization*. [Online]. Available: https://www.sandia.gov/plato3d/index.html. [Accessed: 14-Nov-2020].

10 Appendix

A Extra Figures

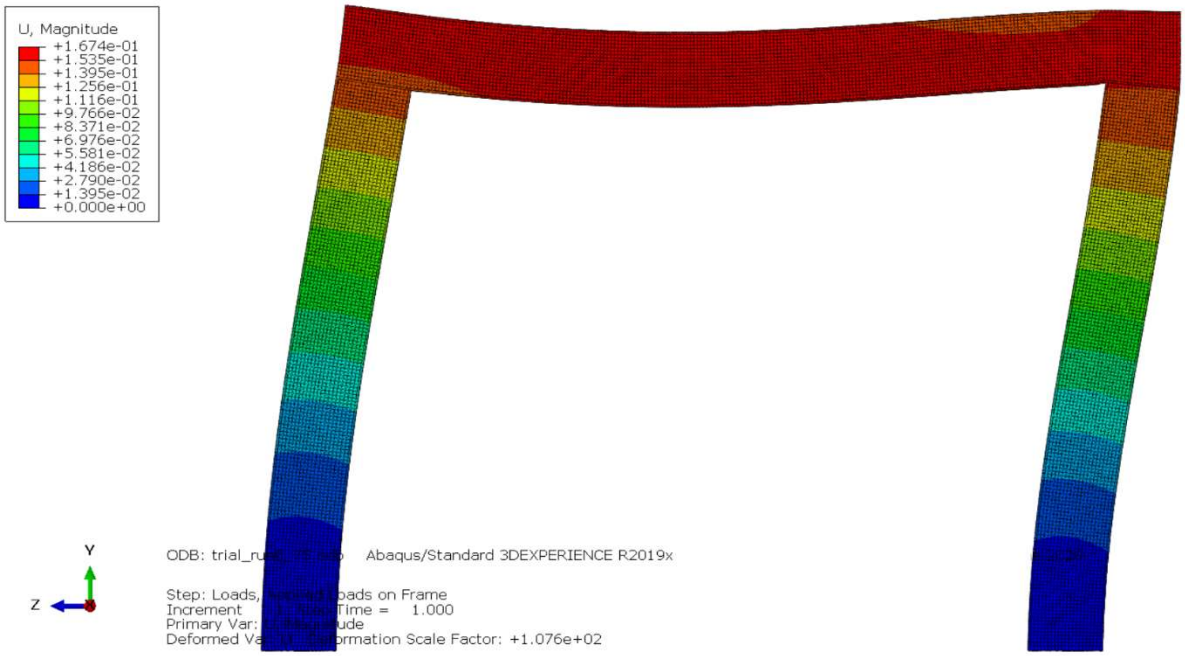

Figure A1: Displacement magnitude plot on the converged mesh. Values in psi.

Figure A2: Side view of Figure 7.

Figure A3: Topology optimization of the frame with only applied gravity loading.

Figure A4: Side view of Figure A3.

Figure A5: Topology optimization of the frame with only applied gravity loading.

Figure A6: Side view of Figure A5.

B Extra Tables

Mesh	Seed Size	Number of Elements	Number of Nodes	Max Vertical Displacement	% Error
1	48	152	1177	-0.0337	31.03
2	24	192	1482	-0.0438	10.25
3	12	320	2458	-0.0445	8.79
4	6	584	4471	-0.0457	6.52
5	3	1642	12247	-0.0463	5.24
6	1.5	5740	41658	-0.0465	4.74
7	0.75	21507	153472	-0.0466	4.53

Table B1: Percent error for maximum vertical displacement.

Mesh	Seed Size	Number of Elements	Number of Nodes	Max Horizontal Displacement	% Error
1	48	152	1177	-1.44E-01	16.32
2	24	192	1482	-1.60E-01	29.27
3	12	320	2458	-1.60E-01	29.48
4	6	584	4471	-1.63E-01	31.99
5	3	1642	12247	-1.66E-01	33.93
6	1.5	5740	41658	-1.67E-01	34.75
7	0.75	21507	153472	-1.67E-01	35.10

Table B2: Percent error for maximum vertical displacement.

C Loading Calculation

Gravity Load:

$$DL = 1{,}000\,plf \rightarrow 83.3\,pli$$

$$P_{DL,gmsh} = \frac{DL}{b_{flange}} \rightarrow 15.1\,psi \quad \text{For GMSH geometry}$$

$$P_{DL,abaqus} = \frac{DL}{t_{web}} \rightarrow 303\,psi \quad \text{For ABAQUS geometry}$$

Earthquake Load:

$$EL = 10\,kips \rightarrow 10{,}000\,lb$$

$$P_{EL,gmsh} = \frac{EL}{b_{flange} \cdot L_b} \rightarrow 10\,psi \quad \text{For GMSH geometry}$$

$$P_{EL,abaqus} = \frac{EL}{t_{web} \cdot b_{depth}} \rightarrow 2{,}287\,psi \quad \text{For ABAQUS geometry}$$

D Python Code

```
# -*- coding: mbcs -*-
# Do not delete the following import lines
from abaqus import *
from abaqusConstants import *
import __main__
import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior
import os
import shutil

'''
PROGRAM TO CREATE, MESH, AND RUN JOB FOR W-SHAPE PORTAL FRAME
SUPPORT CONDITIONS: FIXED
LOAD TYPE = GRAVITY AND EARTHQUAKE
'''

# For W16x31 - Shape Properties - Units: Inches
# Beam
b_length = 180  # Beam length
b_depth = 15.9  # Beam depth, d
b_fwidth = 5.53  # Beam flange width, bf
b_fthick = 0.440  # Beam flange thickness, tf
b_wthick = 0.275  # Beam web thickness, tw
# Column
c_length = 120  # Column length
c_depth = 15.9  # Column depth, d
c_fwidth = 5.53  # Column flange width, bf
c_fthick = 0.440  # Column flange thickness, tf
c_wthick = 0.275  # Column web thickness, tw

# Loads - Pressures - Units: PSI
grav_load = 303.  # Gravity load on beam top flange
eq_load = 2287.  # Earthquake load on side of beam web

# Job Run Parameters
seed_no = [48, 24, 12, 6, 3, 1.5, 0.75]  # Number of global seeds
analysis_enviro = 1  # Use 1 for local machine, 2 for remote machine
job_name = 'trial_run'  # Folder must already be created with this name
```

```
# Environment Selection
if analysis_enviro == 1:  # Corresponds to local machine
    results_dir = 'C:/Users/Jose Vera/Documents/College/Fall 2020/Finite
Elements' \
                  '/PROJECT/Python-Abaqus Interface/'
    home_dir = 'C:/Users/Jose Vera/Documents/College/ABAQUS/'
elif analysis_enviro == 2:  # Corresponds to remote UofU Finite Elements Pool
0
    results_dir = 'X:/Documents/'
    home_dir = 'C:/temp/'



'''W-SHAPE PART CREATION'''

'''SECTIONING'''
# Beam
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
                                             sheetSize=200.0)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
# Set construction lines for mirror
s1.setPrimaryObject(option=STANDALONE)
s1.ConstructionLine(point1=(0.0, 0.0), angle=90.0)
s1.VerticalConstraint(entity=g[2], addUndoState=False)
s1.ConstructionLine(point1=(0.0, 0.0), angle=0.0)
s1.HorizontalConstraint(entity=g[3], addUndoState=False)
# Draw first quadrant of section
s1.setPrimaryObject(option=STANDALONE)
s1.Line(point1=(b_wthick / 2, b_depth / 2), point2=(b_fwidth / 2, b_depth /
2))
s1.HorizontalConstraint(entity=g[4], addUndoState=False)
s1.Line(point1=(b_fwidth / 2, b_depth / 2), point2=(b_fwidth / 2, b_depth / 2
- b_fthick))
s1.VerticalConstraint(entity=g[5], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)
s1.Line(point1=(b_fwidth / 2, b_depth / 2 - b_fthick), point2=(b_wthick / 2,
b_depth / 2 - b_fthick))
s1.HorizontalConstraint(entity=g[6], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[5], entity2=g[6], addUndoState=False)
s1.Line(point1=(b_wthick / 2, b_depth / 2 - b_fthick), point2=(b_wthick / 2,
0.0))
s1.VerticalConstraint(entity=g[7], addUndoState=False)
s1.PerpendicularConstraint(entity1=g[6], entity2=g[7], addUndoState=False)
# Mirror operations
s1.copyMirror(mirrorLine=g[2], objectList=(g[4], g[5], g[6], g[7]))
s1.Line(point1=(-b_wthick / 2, b_depth / 2), point2=(b_wthick / 2, b_depth /
2))
s1.HorizontalConstraint(entity=g[12], addUndoState=False)
s1.copyMirror(mirrorLine=g[3], objectList=(g[4], g[5], g[6], g[7],
                                           g[8], g[9], g[10], g[11], g[12]))
# Part parameter assignments
p = mdb.models['Model-1'].Part(name='BEAM', dimensionality=THREE_D,
                               type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['BEAM']
p.BaseSolidExtrude(sketch=s1, depth=b_length)

# Column
```

```
s2 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
                                             sheetSize=200.0)
g, v, d, c = s2.geometry, s2.vertices, s2.dimensions, s2.constraints
# Set construction lines for mirror
s2.setPrimaryObject(option=STANDALONE)
s2.ConstructionLine(point1=(0.0, 0.0), angle=90.0)
s2.VerticalConstraint(entity=g[2], addUndoState=False)
s2.ConstructionLine(point1=(0.0, 0.0), angle=0.0)
s2.HorizontalConstraint(entity=g[3], addUndoState=False)
# Draw first quadrant of section
s2.setPrimaryObject(option=STANDALONE)
s2.Line(point1=(b_wthick / 2, b_depth / 2), point2=(b_fwidth / 2, b_depth /
2))
s2.HorizontalConstraint(entity=g[4], addUndoState=False)
s2.Line(point1=(b_fwidth / 2, b_depth / 2), point2=(b_fwidth / 2, b_depth / 2
- b_fthick))
s2.VerticalConstraint(entity=g[5], addUndoState=False)
s2.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)
s2.Line(point1=(b_fwidth / 2, b_depth / 2 - b_fthick), point2=(b_wthick / 2,
b_depth / 2 - b_fthick))
s2.HorizontalConstraint(entity=g[6], addUndoState=False)
s2.PerpendicularConstraint(entity1=g[5], entity2=g[6], addUndoState=False)
s2.Line(point1=(b_wthick / 2, b_depth / 2 - b_fthick), point2=(b_wthick / 2,
0.0))
s2.VerticalConstraint(entity=g[7], addUndoState=False)
s2.PerpendicularConstraint(entity1=g[6], entity2=g[7], addUndoState=False)
# Mirror operations
s2.copyMirror(mirrorLine=g[2], objectList=(g[4], g[5], g[6], g[7]))
s2.Line(point1=(-b_wthick / 2, b_depth / 2), point2=(b_wthick / 2, b_depth /
2))
s2.HorizontalConstraint(entity=g[12], addUndoState=False)
s2.copyMirror(mirrorLine=g[3], objectList=(g[4], g[5], g[6], g[7],
                                           g[8], g[9], g[10], g[11], g[12]))
# Part parameter assignments
p = mdb.models['Model-1'].Part(name='COLUMN', dimensionality=THREE_D,
                               type=DEFORMABLE_BODY)
p.BaseSolidExtrude(sketch=s2, depth=c_length)

# Base Plate
s3 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
                                             sheetSize=200.0)
g, v, d, c = s3.geometry, s3.vertices, s3.dimensions, s3.constraints
s3.setPrimaryObject(option=STANDALONE)
s3.Line(point1=(-c_fwidth / 2, -c_depth / 2), point2=(c_fwidth / 2, -c_depth
/ 2))
s3.HorizontalConstraint(entity=g[2], addUndoState=False)
s3.Line(point1=(c_fwidth / 2, -c_depth / 2), point2=(c_fwidth / 2, c_depth /
2))
s3.VerticalConstraint(entity=g[3], addUndoState=False)
s3.PerpendicularConstraint(entity1=g[2], entity2=g[3], addUndoState=False)
s3.Line(point1=(c_fwidth / 2, c_depth / 2), point2=(-c_fwidth / 2, c_depth /
2))
s3.HorizontalConstraint(entity=g[4], addUndoState=False)
s3.PerpendicularConstraint(entity1=g[3], entity2=g[4], addUndoState=False)
s3.Line(point1=(-c_fwidth / 2, c_depth / 2), point2=(-c_fwidth / 2, -c_depth
/ 2))
s3.VerticalConstraint(entity=g[5], addUndoState=False)
```

```
s3.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)
s3.PerpendicularConstraint(entity1=g[2], entity2=g[5], addUndoState=False)
# Part parameter assignments
p = mdb.models['Model-1'].Part(name='BASE-PLATE', dimensionality=THREE_D,
                               type=DEFORMABLE_BODY)
p.BaseSolidExtrude(sketch=s3, depth=1.0)

'''MATERIAL CREATION'''
# STEEL
mdb.models['Model-1'].Material(name='Steel')
mdb.models['Model-1'].materials['Steel'].Elastic(table=((29000000.0,
0.303),))

'''SECTION ASSIGNMENT'''
# Beam
mdb.models['Model-1'].HomogeneousSolidSection(name='Beam-Section',
                                              material='Steel',
thickness=None)
p = mdb.models['Model-1'].parts['BEAM']
c = p.cells
cells = c.getSequenceFromMask(mask=('[#1f ]',), )
region = regionToolset.Region(cells=cells)
p.SectionAssignment(region=region, sectionName='Beam-Section', offset=0.0,
                    offsetType=MIDDLE_SURFACE, offsetField='',
                    thicknessAssignment=FROM_SECTION)
# Column
mdb.models['Model-1'].HomogeneousSolidSection(name='Column-Section',
                                              material='Steel',
thickness=None)
p = mdb.models['Model-1'].parts['COLUMN']
c = p.cells
cells = c.getSequenceFromMask(mask=('[#1f ]',), )
region = regionToolset.Region(cells=cells)
p.SectionAssignment(region=region, sectionName='Column-Section', offset=0.0,
                    offsetType=MIDDLE_SURFACE, offsetField='',
                    thicknessAssignment=FROM_SECTION)
# Base Plate
mdb.models['Model-1'].HomogeneousSolidSection(name='BP-Section',
                                              material='Steel',
thickness=None)
p = mdb.models['Model-1'].parts['BASE-PLATE']
c = p.cells
cells = c.getSequenceFromMask(mask=('[#1f ]',), )
region = regionToolset.Region(cells=cells)
p.SectionAssignment(region=region, sectionName='BP-Section', offset=0.0,
                    offsetType=MIDDLE_SURFACE, offsetField='',
                    thicknessAssignment=FROM_SECTION)

'''PARTITIONING'''
p = mdb.models['Model-1'].parts['BEAM']
c = p.cells
pickedCells = c.getSequenceFromMask(mask=('[#1 ]',), )
f = p.faces
p.PartitionCellByExtendFace(extendFace=f[3], cells=pickedCells)
c = p.cells
pickedCells = c.getSequenceFromMask(mask=('[#7 ]',), )
f1 = p.faces
```

```
p.PartitionCellByExtendFace(extendFace=f1[17], cells=pickedCells)

p = mdb.models['Model-1'].parts['COLUMN']
c = p.cells
pickedCells = c.getSequenceFromMask(mask=('[#1 ]',), )
f = p.faces
p.PartitionCellByExtendFace(extendFace=f[3], cells=pickedCells)
c = p.cells
pickedCells = c.getSequenceFromMask(mask=('[#7 ]',), )
f1 = p.faces
p.PartitionCellByExtendFace(extendFace=f1[17], cells=pickedCells)

'''INSTANCES'''
# Create Instances
a1 = mdb.models['Model-1'].rootAssembly
p = mdb.models['Model-1'].parts['BEAM']
a1.Instance(name='BEAM-1', part=p, dependent=ON)
p = mdb.models['Model-1'].parts['COLUMN']
a1.Instance(name='COLUMN-1', part=p, dependent=ON)
p = mdb.models['Model-1'].parts['COLUMN']
a1.Instance(name='COLUMN-2', part=p, dependent=ON)
p = mdb.models['Model-1'].parts['BASE-PLATE']
a1.Instance(name='BASE-PLATE-1', part=p, dependent=ON)
p = mdb.models['Model-1'].parts['BASE-PLATE']
a1.Instance(name='BASE-PLATE-2', part=p, dependent=ON)
# Assemble Frame Geometry
a1.rotate(instanceList=('COLUMN-1', 'COLUMN-2',), axisPoint=(1.0, 0.0, 0.0),
          axisDirection=(-1.0, 0.0, 0.0), angle=-90.0)
a1.translate(instanceList=('COLUMN-1',), vector=(0.0, -(b_depth / 2),
(c_depth / 2)))
a1.translate(instanceList=('COLUMN-2',), vector=(0.0, -(b_depth / 2),
b_length - (c_depth / 2)))
a1.rotate(instanceList=('BASE-PLATE-1', 'BASE-PLATE-2',), axisPoint=(1.0, -
(b_depth / 2), 0.0),
          axisDirection=(-1.0, 0.0, 0.0), angle=-90.0)
a1.translate(instanceList=('BASE-PLATE-1',), vector=(0.0, -c_length, 0.0))
a1.translate(instanceList=('BASE-PLATE-2',), vector=(0.0, -c_length, b_length
- (c_depth)))
# Merge geometries to create new part FRAME
a1.InstanceFromBooleanMerge(name='FRAME', instances=(a1.instances['BEAM-1'],
                                                     a1.instances['COLUMN-
1'], a1.instances['COLUMN-2'],
                                                     a1.instances['BASE-
PLATE-1'], a1.instances['BASE-PLATE-2'],),
                            keepIntersections=ON, originalInstances=SUPPRESS,
domain=GEOMETRY)

'''BOUNDARY CONDITIONS'''
# Fixed Supports
a = mdb.models['Model-1'].rootAssembly
f1 = a.instances['FRAME-1'].faces
faces1 = f1.getSequenceFromMask(mask=('[#1020 ]',), )
region = regionToolset.Region(faces=faces1)
mdb.models['Model-1'].DisplacementBC(name='FIXED', createStepName='Initial',
                                     region=region, u1=SET, u2=SET, u3=SET,
ur1=SET, ur2=SET, ur3=SET,
                                     amplitude=UNSET,
```

```
distributionType=UNIFORM, fieldName='',
                                        localCsys=None)
# Loads on Frame
# Gravity load on beam
mdb.models['Model-1'].StaticStep(name='Loads', previous='Initial',
                                   description='Applied Loads on Frame')
s1 = a.instances['FRAME-1'].faces
side1Faces1 = s1.getSequenceFromMask(mask=('[#0:2 #20000000 ]',), )
region = regionToolset.Region(side1Faces=side1Faces1)
mdb.models['Model-1'].Pressure(name='Applied Pressure',
createStepName='Loads',
                                region=region, distributionType=UNIFORM,
field='', magnitude=grav_load,
                                amplitude=UNSET)
# Earthquake load on side of beam
s1 = a.instances['FRAME-1'].faces
side1Faces1 = s1.getSequenceFromMask(mask=('[#0:2 #4000000 ]',), )
region = regionToolset.Region(side1Faces=side1Faces1)
mdb.models['Model-1'].Pressure(name='EQ', createStepName='Loads',
                                region=region, distributionType=UNIFORM,
field='', magnitude=eq_load,
                                amplitude=UNSET)

'''FOR LOOP FOR MULTIPLE JOB CREATION WITH DIFFERENT CONDITIONS'''
# Condition to change: Global seeds

for i in range(len(seed_no)):

    '''SEEDING'''
    p = mdb.models['Model-1'].parts['FRAME']
    e = p.edges
    pickedEdges = e.getSequenceFromMask(mask=('[#ffffffff:6 #3f ]',), )
    p.seedEdgeBySize(edges=pickedEdges, size=seed_no[i], deviationFactor=0.1)
    elemType1 = mesh.ElemType(elemCode=C3D20R, elemLibrary=STANDARD)
    elemType2 = mesh.ElemType(elemCode=C3D15, elemLibrary=STANDARD)
    elemType3 = mesh.ElemType(elemCode=C3D10, elemLibrary=STANDARD,
                              secondOrderAccuracy=OFF,
distortionControl=DEFAULT)
    c = p.cells
    cells = c.getSequenceFromMask(mask=('[#1ffff ]',), )
    pickedRegions = (cells,)
    # Set element type to quadratic tetrahedron
    if i == 0:
        p.setElementType(regions=pickedRegions, elemTypes=(elemType1,
elemType2,
                                                           elemType3))
    p.generateMesh()
    # Replace period in decimal with "_"
    if seed_no[i] < 2:
        seed_str = str(seed_no[i]).replace(".", "_")
    else:
        seed_str = str(seed_no[i])

    '''JOB CREATION'''
    mdb.Job(name=job_name + seed_str, model='Model-1', description='',
type=ANALYSIS,
            atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
```

```
            memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
            explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE,
echoPrint=OFF,
            modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
userSubroutine='',
            scratch='', resultsFormat=ODB)
    mdb.jobs[job_name + seed_str].submit(consistencyChecking=OFF)
    mdb.jobs[job_name + seed_str].waitForCompletion()

    '''POST-PROCESSING'''
    # Set viewport zoom and perspective
    jobPath = job_name + seed_str + '.odb'
    odb_object = session.openOdb(name=jobPath)
    session.viewports['Viewport: 1'].setValues(displayedObject=odb_object)
    session.viewports['Viewport: 1'].view.setViewpoint(viewVector=(1, 0, 0),
                                                       cameraUpVector=(0, 1,
0))
    session.viewports['Viewport: 1'].view.fitView()
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
        DEFORMED,))
    session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
        CONTOURS_ON_DEF,))
    session.printOptions.setValues(reduceColors=False)
    # Print SMises plot on deformed shape
    session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='S', outputPosition=INTEGRATION_POINT, refinement=(
            INVARIANT, 'Mises'), )
    session.printToFile(
        fileName=results_dir + job_name + '/SMises_Deformed' + seed_str,
        format=TIFF, canvasObjects=(session.viewports['Viewport: 1'],))
    # Print displacement plot on deformed shape
    session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='U', outputPosition=NODAL, refinement=(INVARIANT,
                                                             'Magnitude'), )
    session.printToFile(
        fileName=results_dir + job_name + '/Displacements_Deformed' +
seed_str,
        format=TIFF, canvasObjects=(session.viewports['Viewport: 1'],))
    # Print report with SMises and displacements
    odb = session.odbs[home_dir
                       + job_name + seed_str + '.odb']
    session.writeFieldReport(fileName=job_name + seed_str + '.rpt',
append=ON,
                             sortItem='Node Label', odb=odb, step=0, frame=1,
outputPosition=NODAL,
                             variable=(('U', NODAL, ((COMPONENT, 'U2'), (
                                 COMPONENT, 'U3'),)), ('S',
INTEGRATION_POINT,
                                                       ((INVARIANT,
'Mises'),)),), stepFrame=SPECIFY)
    shutil.copy(home_dir + job_name + seed_str + '.rpt', results_dir +
job_name + '/')
```

E PLATO Input Code

```
begin objective
   type maximize stiffness
   load ids 1 2
   boundary condition ids 1 2
   code plato_analyze
   number processors 1
end objective

begin boundary conditions
   fixed displacement sideset name sideset_1 bc id 1
   fixed displacement sideset name sideset_2 bc id 2
end boundary conditions

begin loads
    traction sideset name sideset_4 value 0 -15 0 load id 1
    traction sideset name sideset_4 value 0 0 -10 load id 2
end loads

begin constraint
   type volume
   volume fraction .35
end constraint

begin block 1
   material 1
end block

begin material 1
   poissons ratio .303
   youngs modulus 29e6
end material

begin optimization parameters
   number processors 1
   filter radius scale 2.5
   max iterations 2000
   output frequency 1000
   algorithm oc
   discretization density
   initial density value .5
end optimization parameters

begin mesh
   name frame.exo
end mesh

begin code
   code PlatoMain
   code analyze_MPMD
end code
```