SANDIA REPORT
SAND2019-12111
Printed October, 2019

Sandia National Laboratories

Simulations of Pipe Overpack Container Compaction at the Waste Isolation Pilot Plant

Benjamin Reedlunn, James Bean

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

ABSTRACT

Nuclear criticality experts at Oak Ridge National Laboratories (ORNL) are currently analyzing 6-inch and 12-inch Standard Pipe Overpack (POP) containers after disposal at the Waste Isolation Pilot Plant (WIPP). POP containers are emplaced in WIPP disposal rooms that will naturally close and compact their contents closer to one another over several centuries. This report details a few simulations that predict the final compacted container configurations as inputs to ORNL's criticality studies. Each POP container was discretely modeled, including the plywood, stainless steel pipe, and fiberboard inside the 55-gallon drum, in order to capture the container's complex mechanical behavior and interactions between neighboring containers. A roof fall compaction simulation demonstrated that both 6-inch and 12-inch Standard POP containers are unlikely to substantially deform or scatter if a large block of salt detaches from a disposal room ceiling, free falls, and lands almost flat on the containers. Gradual compaction simulations predicted the compacted shape 1000 yr after container emplacement. Tracking of the stainless steel pipe centroids found the minimum and mean predicted distances between any two 6-inch pipe centroids were 0.109 m and 0.237 m, respectively, while the minimum and mean distances between 12-inch pipe centroids were 0.233 m and 0.315 m, respectively.

ACKNOWLEDGMENT

The authors would like to acknowledge Brad Day for his helpful suggestions as the work was being performed. Courtney Herrick and Moo Lee also provided useful comments upon reviewing a draft of the report.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

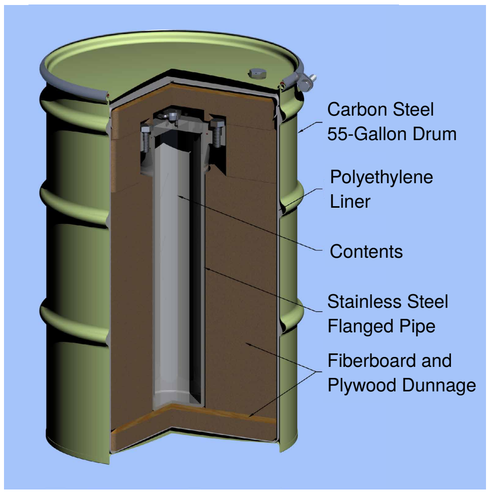
1. INTRODUCTION

The Waste Isolation Pilot Plant (WIPP) is an operating geologic repository in southeastern New Mexico for transuranic (TRU) waste from nuclear defense activities. Past nuclear criticality concerns have generally been low at the WIPP due to the low initial concentration of fissile material and the natural tendency of fissile solute to disperse during fluid transport in porous media (Rechard et al. 2000). On the other hand, the list of acceptable WIPP waste types has expanded over the years to include Criticality Control Overpack (CCO) containers and Pipe Overpack (POP) containers. Containers bound for WIPP are bundled together in hexagon shaped 7-packs (six containers surround one container in the center). Two 7-packs are often combined into a TRUPACT-II package for a total of 14 containers. Most TRUPACT-II packages are restricted to a maximum fissile mass equivalent to plutonium (FMEP) between 0.1 and 0.38 kg, but a CCO TRUPACT-II package and a POP TRUPACT-II package are respectively permitted to have 5.32 kg and 2.80 kg FMEP (see Section 3 of US DOE (2013)). Consequently, CCO container criticality after emplacement at the WIPP was evaluated in Saylor and Scaglione (2018), and Oak Ridge National Laboratories is currently at work on POP container criticality analyses.

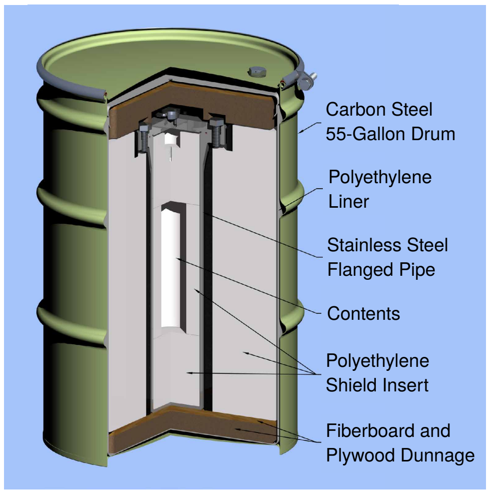
Once waste containers are placed in an underground disposal room at the WIPP, the natural process of room closure will compress them over centuries. This compaction will have an impact on nuclear criticality assessments, so it is important to have predictions of the final compacted container configuration. Reedlunn et al. (2019) predicted the gradual compaction of CCO containers and this report documents similar gradual POP container compaction predictions.

Chapter 1 of Reedlunn et al. (2019) discussed one significant caveat to the modeling approach pursued therein: the gradual compaction simulations did not include the effect of roof falls. If left unrestrained, fractures above a room can cause large blocks of salt to fall into a room. Rock falls could potentially crush and scatter containers, leading to clusters of containers (high equivalent plutonium density) in some regions and large regions of salt between containers (low equivalent plutonium density) in other regions. The gradual compaction simulations in Reedlunn et al. (2019) are still useful estimates of CCO container compaction averaged over the volume of the room, but container clustering could be important for criticality assessments. We, therefore, developed a separate roof fall compaction model herein to assess the likelihood of POP container clustering prior to gradual compaction.

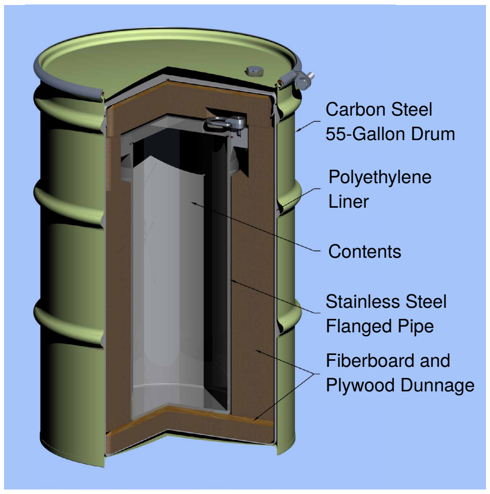
Six different POP container designs exist (Porter 2013), but only the 6-inch and 12-inch Standard diameter pipe design in Fig. 1-1c were analyzed herein. As shown in Fig. 1-1, the 6-inch Standard and S100 designs include a 6-inch diameter stainless steel flanged pipe, while the 12-inch Standard, S200-A, S200-B, and S300 include a 12-inch diameter stainless steel flanged pipe. These pipes are the most significant structural feature in each design. We chose not to analyze the S200-A, S200-B, and S300 because they all are stiffer than the 12-inch Standard design due to the materials inside the 12-inch diameter pipe. We also chose not to analyze the S100 because it is



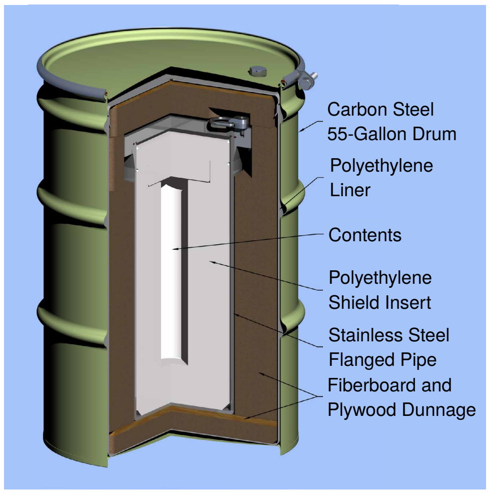
(a) 6-inch Standard



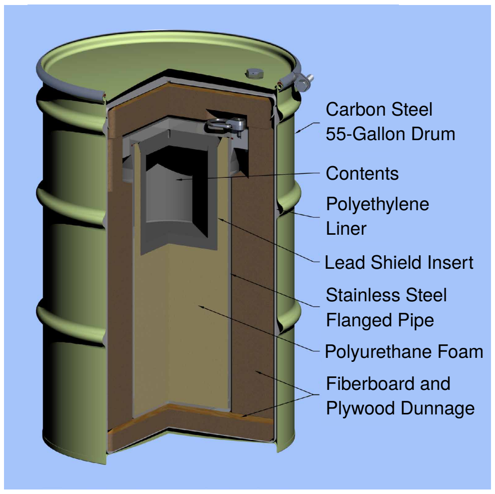
(b) S100



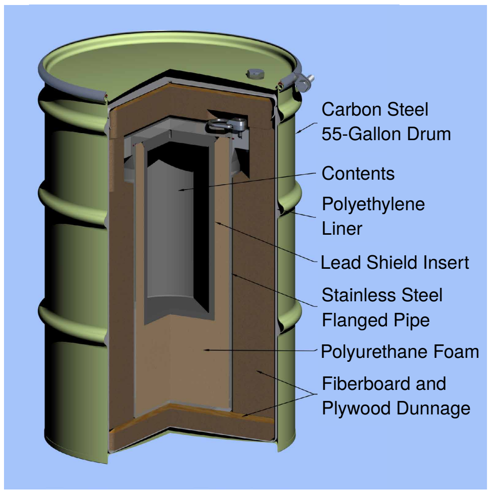
(c) 12-inch Standard



(d) S300



(e) S200-A



(f) S200-B

Figure 1-1. Pipe-Overpack container designs (quarter wedges removed for visualization purposes) (modified from Porter (2013)).

only used for fissile material in the form of small capsules that cannot consolidate to a more reactive geometry (Day 2019). This leaves the 6-inch Standard and 12-inch Standard designs.

We chose to analyze both the 6-inch and 12-inch designs because it was not immediately obvious which design would be more reactive after being fully compacted by room closure. The 6-inch pipe has an outer diameter D_p to thickness B_p ratio of $D_\mathrm{p}/B_\mathrm{p} = 23.7$ and a length L_p to diameter ratio of $L_\mathrm{p}/D_\mathrm{p} = 3.9$. The 12-inch pipe has non-dimensional ratios of $D_\mathrm{p}/B_\mathrm{p} = 51.0$ and $L_\mathrm{p}/D_\mathrm{p} = 2.0$. These ratios indicate the 6-inch pipe is more susceptible to column buckling and the 12-inch pipe is more susceptible to shell buckling, but it is not immediately evident which buckling mode will dominate as the room closure compacts the containers. If both pipes buckle similarly, then the 6-inch pipes would likely be more compressible simply because they contain less stainless steel to resist room closure than the 12-inch pipes. The 12-inch pipes, on the other hand, have more volume for waste storage, which means an optimally moderated hydrogen to plutonium ratio is more plausible inside 12-inch pipes. In other words, even if the 6-inch Standard containers compress further, the 12-inch Standard container could still be more reactive (Day 2019). One might be tempted to focus exclusively on the 12-inch Standard containers, because 97 % of the POP containers emplaced at the WIPP are 12-inch Standard containers, while the 6-inch Standard container has yet to be used, as of June 2019 (Kirkes 2019). On the other hand, a waste generation site may wish to use 6-inch Standard containers in the future, and removing the design from the list of acceptable waste forms is not a trivial procedure. In conclusion, compaction analyses of both 6-inch and 12-inch designs are needed.

The gradual compaction of 6-inch and 12-inch Standard POP containers due to WIPP room closure has been simulated before. Park and Hansen (2004) fit an isotropic volumetric plasticity model to the simulated compaction behavior of a single POP container of each design. Park and Hansen (2005) then assumed the homogenized behavior of one container was the same as the behavior of the hundreds of containers of the same design in a room. Six images from Park and Hansen (2005) simulations involving 6-inch and 12-inch Standard POP containers are shown in Fig. 1-2. The left side of each image is a mirror boundary condition and the gas pressure from decomposing waste was assumed to be zero. The salt first comes into contact with the waste containers at 25 years and slowly compresses them over the next 10,000 years. The homogenized waste container deformation is not severe and remains relatively uniform throughout the compaction process. The Park and Hansen (2004) and Park and Hansen (2005) approach was reasonable given the computational resources at the time, but it likely produced overly stiff waste behavior. (See Chapter 1 of Reedlunn et al. (2019) for further discussion.)

In this work, we elected to model each container, and its components, discretely to more accurately capture the container stiffness. This discrete approach was originally developed in Reedlunn et al. (2019) for the gradual compaction of CCO containers, and the same general approach was applied herein to 6-inch and 12-inch Standard POP containers.

Chapter 2 discusses the model setup, Chapter 3 presents the simulation results, and Chapter 4 summarizes the work and lists potential areas for improvement.

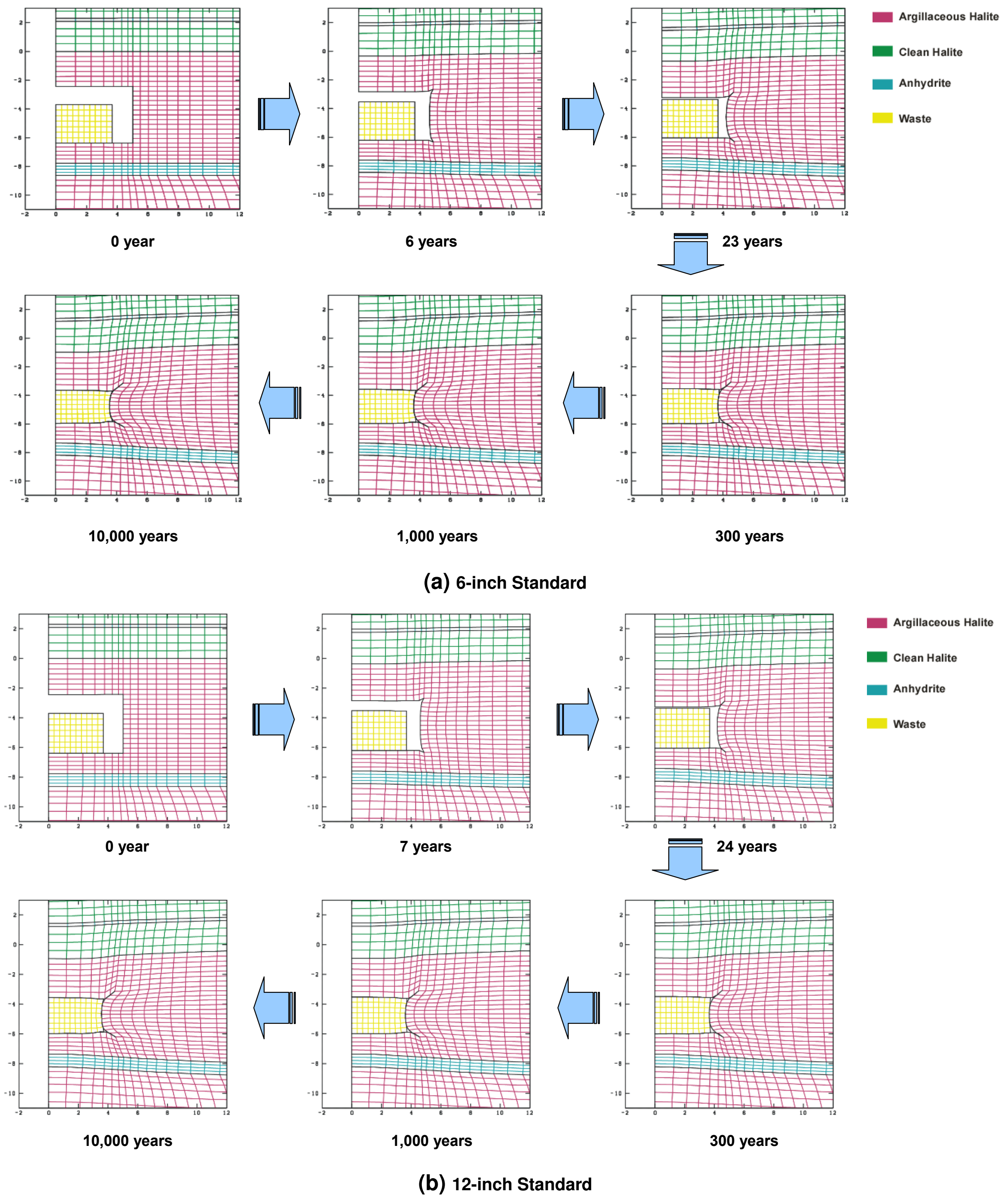


Figure 1-2. Park and Hansen (2005)'s predictions of 6-inch and 12-inch Standard POP container compaction using a homogenized waste model and no gas generation. The left side is a mirror boundary condition. (Fig. 23 and 24 from Park and Hansen (2005))

2. MODEL SETUP

The gradual compaction model (Section 2.1) and roof fall compaction model (Section 2.2) are presented in the order they were developed, followed by a brief description of the numerical approach (Section 2.3).

2.1. Gradual Compaction Model

Sandia's model for gradual room closure at the WIPP has evolved considerably over the years. See Section 2.1.1 of Reedlunn et al. (2019) for a short review. Most recently, Appendix B in Reedlunn et al. (2019) compared the current model predictions against closure measurements in three different shaped empty rooms. The model under-predicted the first year of rapid closure in each case by as much as 2×, but the model predicted the horizontal and vertical closure rates in the ensuing 3 to 7 years within 1.36×. Although more accurately predicting the initial transient would engender further confidence, closure rates after the initial transient are typically considered most important for long-term closure predictions. Thus, the model validation was considered sufficient to predict the final compacted shape of waste containers.

2.1.1. Geomechanical Model

The geomechanical portion of the gradual compaction model was very similar to the model described in Section 2.1 of Reedlunn et al. (2019). Figure 2-1 depicts the model geometry and boundary conditions. The material models for the salt, anhydrite, polyhalite, and sliding interfaces were all defined in Section 2.1 of Reedlunn et al. (2019), along with values of p_{top}, D_{h}, D_{v1}, D_{v2}, L_{h}, and L_{v}. Although not shown in Fig. 2-1, the depth into the page L_{d} was also the same as that employed by Reedlunn et al. (2019). The distance from the top of the room to clay seam G, however, differed in Reedlunn et al. (2019). Reedlunn et al. (2019) modeled a disposal room on the WIPP upper horizon (panels 3, 4, 5, and 6), whereas herein we chose to model a disposal room on the WIPP lower horizon (panels 1, 2, 7, and 8). The top of upper horizon disposal rooms coincide with clay seam G, such that $L_{\mathrm{G}} = 0$, whereas the top of lower horizon disposal rooms are $L_{\mathrm{G}} = 2.43$ m beneath clay seam G. We switched to the lower horizon to maximize the thickness of a potential roof fall block, as discussed in Section 2.2. See Appendix A.4.1 for the Cubit (Cubit Team 2017) journal file used to build the geomechanical finite element mesh and see Appendix A.3 for an image and description of the mesh.

Past simulations of container compaction, such as those in Stone (1997), Park and Hansen (2005), and Park and Holland (2007), included the impact of gas generated from decomposing waste as a pressure boundary condition on the interior of the room. They varied the pressure history profiles

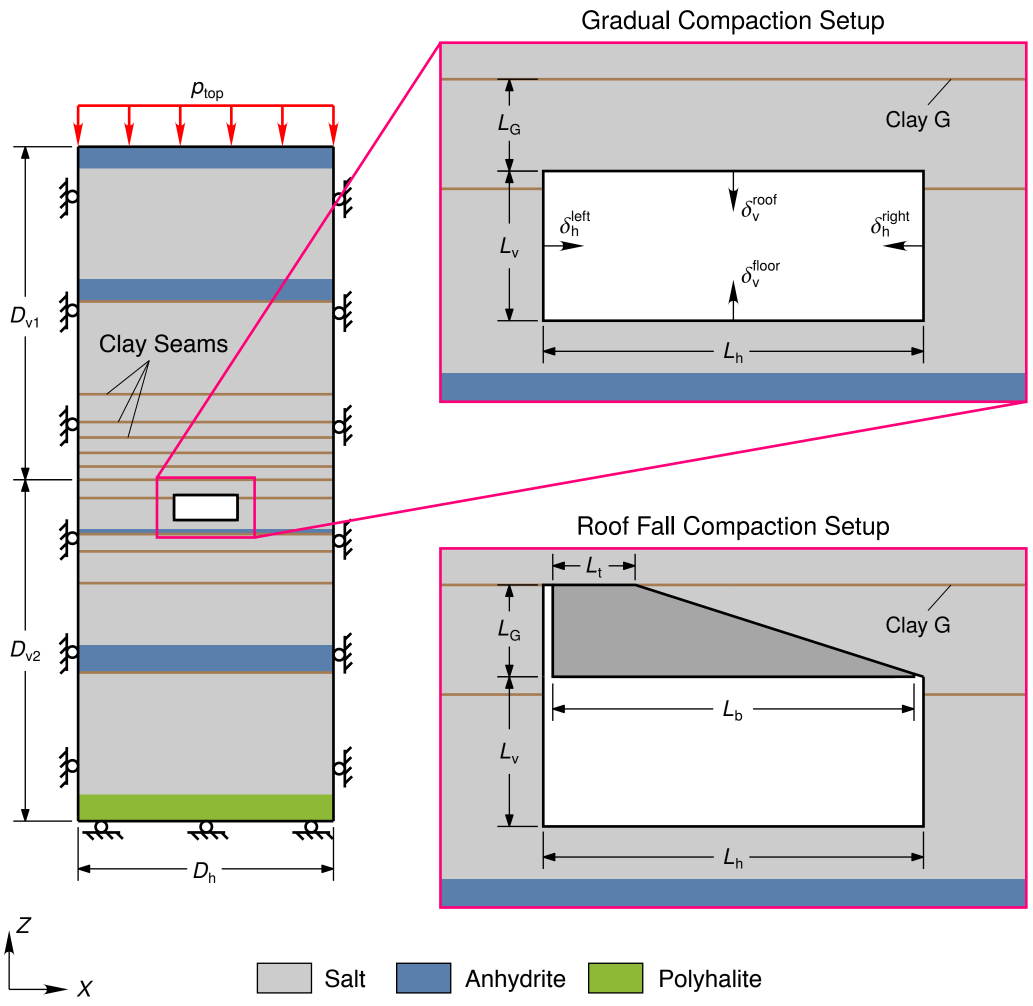


Figure 2-1. Lower horizon geomechanical model geometry and boundary conditions (shown without containers).

over a wide range to capture the uncertainty in the rate of gas generation and whether gas could escape to other parts of the repository. Gas generation was not included in this work in the interest of conservatism. Gas pressure restrains room closure, which would reduce container compaction, and neglecting gas pressure should not significantly affect the room vertical-to-horizontal closure ratio δ_v/δ_h.

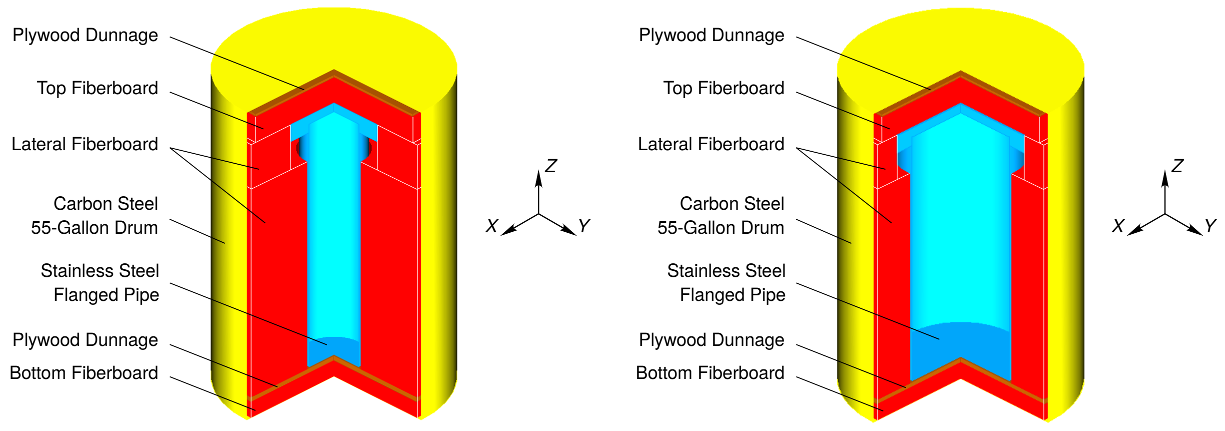
The salt surrounding the room begins the simulation with an initial stress state that is not known a-priori. We can, however, assume the stress state was lithostatic prior to room excavation. (Given enough time, salt's viscoplasticity "erases" any non-lithostatic stress state.) Rather than delete elements to expose the room within the geology, a fluid pressure equal to the lithostatic pressure was first applied to the walls of the room at time $t = -10$ ms, and then the pressure sinusoidally reduced to zero over 10 ms to simulate an instantaneous room excavation. For $t > 0$ the fluid pressure was held to zero so that the room could naturally close in on the containers. (See Section 2.1.2 of Reedlunn et al. (2019) for further details.)

One way to monitor the closure of a disposal room is the horizontal and vertical closure at room mid-height and mid-width, respectively. The magnified view of the gradual compaction setup in Fig. 2-1 defines the horizontal displacement of the left wall $\delta_\mathrm{h}^\mathrm{left}$ and right wall $\delta_\mathrm{h}^\mathrm{right}$, as well as the vertical displacement of the floor $\delta_\mathrm{v}^\mathrm{floor}$ and roof $\delta_\mathrm{v}^\mathrm{roof}$. These displacements varied along the along the Y-direction (into the page) when containers were compacted inside the room, so the mean values $\bar{\delta}_\mathrm{h}^\mathrm{left}$, $\bar{\delta}_\mathrm{h}^\mathrm{right}$, $\bar{\delta}_\mathrm{v}^\mathrm{floor}$, and $\bar{\delta}_\mathrm{v}^\mathrm{roof}$ were computed and used to define the (Y-averaged) horizontal and vertical closure of the room as $\delta_\mathrm{h} = \bar{\delta}_\mathrm{h}^\mathrm{left} + \bar{\delta}_\mathrm{h}^\mathrm{right}$ and $\delta_\mathrm{v} = \bar{\delta}_\mathrm{v}^\mathrm{floor} + \bar{\delta}_\mathrm{v}^\mathrm{roof}$.

2.1.2. Container Model

This section describes how the containers were modeled in the gradual compaction simulations. The container geometry and emplacement are discussed first, followed by the container material models.

2.1.2.1. Container Geometry and Emplacement



(a) **6-inch Standard** **(b)** **12-inch Standard**

Figure 2-2. Idealized container geometries (quarter wedge removed for visualization purposes).

Fig. 2-2 depicts the idealized 6-inch and 12-inch Standard container geometries, each with a $D_\mathrm{c} = 0.574$ m overall diameter and $L_\mathrm{c} = 0.870$ m overall height. One can compare the idealized geometries with the schematics of the actual geometries in Figs. 1-1a and 1-1c, or the technical drawings in Figs. A-1 to A-7, to see that the idealized geometries did not include details such as the ribs of the 55-gallon drums or the bolts in the flanged pipes. These features were thought to be insignificant compared to capturing aspects such as the pipe diameter, thickness, and length. The waste itself was conservatively not included in the container model because no requirement exists for how much waste must be placed in a given container. In addition, the idealized container did not include the “Polyethylene Liner” in Fig. 1-1c (referred to as the “POC Rigid Drum Liner” in Fig. A-6) because it is less than 2.3 mm thick. This “rigid” liner should provide negligible compaction resistance compared to the other container components. See the Cubit journal files in

Appendix A.4.1 for precise dimensions of the idealized geometries, and see Appendix A.3 for images and descriptions of the container finite element meshes.

A schematic of the 153 containers upon emplacement in the disposal room is shown in Fig. 2-3. See Section 2.2.1 of Reedlunn et al. (2019) for a more thorough description of the emplacement, but three details are noted here.

1. An unresolved numerical issue prevented placing 36 half-containers at the Y-boundaries of the simulation domain.
2. The $D_c = 0.574$ m diameter containers were placed only 5 mm apart, ignoring the drum ribs and bolt ring.
3. The sacks of magnesium oxide (MgO), which are placed on a portion of the container stacks at the WIPP, were conservatively not included. (See Section 2.2.2 of Reedlunn et al. (2019) for further discussion.)

See the Cubit journal files in Appendix A.4.1 for precise emplacement dimensions.

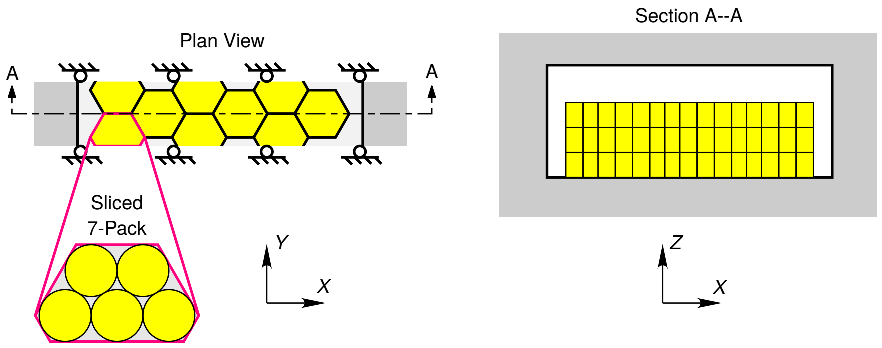


Figure 2-3. Container locations upon emplacement.

2.1.2.2. *Container Material Models*

The materials included in the container model were 304L stainless steel for the flanged pipe, cold rolled commercial quality carbon steel for the 55-gallon drum, exterior grade type AC plywood, and softwood-based Celotex® fiberboard.

The material models employed in Park and Hansen (2004) were initially considered for use herein. Park and Hansen (2004) obtained their material models from Ludwigsen et al. (1998), except Park and Hansen (2004) allowed the plywood to plastically deform, while Ludwigsen et al. (1998) did not. Unfortunately, Ludwigsen et al. (1998) did not indicate how they selected their material parameters. In addition, fracture models were also desired for this work, and Ludwigsen et al. (1998) did not include fracture in their simulations. Therefore, we elected to utilize the stainless steel, carbon steel, and plywood material models calibrated in Section 2.2.2 of Reedlunn et al. (2019). Reedlunn et al. (2019) utilized nominal and strong parameter sets, yet the gradual compaction predictions were relatively insensitive to the stainless steel, carbon steel, and plywood behavior. Therefore, only the nominal "elastic-plastic-fracture" parameter sets were used for the gradual compaction simulations herein. Reedlunn et al. (2019) did not create a fiberboard material model, so a new material model was selected and calibrated.

The fiberboard in the POP containers is primarily used to protect the stainless steel pipe from mechanical impact during transportation, but we elected to include it in our simulations because it could provide some resistance upon sufficient compaction and it holds the stainless steel pipe in place inside the container. According to Walker (1991), Celotex® fiberboard is manufactured in 12.7 mm thick felted layers that are laminated into sheets up to 50.8 mm thick by the manufacturer.

Walker (1991) performed six, room temperature, unconfined, uniaxial compression tests on Celotex® fiberboard samples with initial densities within the range 0.256 ± 0.03 g/cm^3, which matches the density range specified in Figs. A-3 and A-6. Three tests were performed with the loading perpendicular ($\perp$) to the laminations and three were performed parallel ($\|$) to the laminations. The cubic shaped samples were $L_0 = 50.8$ mm on each side. They were either loaded to approximately 222.4 kN or to 90 % reduction in thickness, whichever occurred first. The cross head displacement rate was $\dot{\delta} = -0.04$ mm/s. The test results were presented as plots of load F versus deflection δ, so the load-deflection curves were manually digitized from the figures and converted to engineering stress F/A_0 and engineering strain δ/L_0 using the initial cross-sectional area $A_0 = 2580$ mm^2 and length L_0. As shown in Fig. 2-4, the material produces less than 1 MPa of compressive engineering stress until about $-\delta/L_0 = 40$ %, after which the fiberboard gradually, yet markedly, stiffens. The onset of this lock-up is delayed in the parallel to laminations tests, resulting in lower absolute values of stress for the same strain level. No lateral deformation data was presented by Walker (1991), but Smith and Vormelker (2000) stated that their Celotex® specimens did not laterally bulge during room temperature, laterally unconfined, uniaxial compression. The Smith and Vormelker (2000) study compared reasonably with the data presented by Walker (1991) for the same loading conditions, finding similar levels of anisotropy upon comparing loading perpendicular and parallel to the laminations.

The Flexible Foam constitutive model (Neilsen et al. 2018) was selected to model the fiberboard material. This model can be represented in one-dimension as a Maxwell element (linear elastic spring in series with a dashpot) in parallel with a nonlinear elastic spring. It is designed to capture the load "plateau" as the fiberboard transitions from a porous material (high porosity) to a more solid-like (low porosity) material. It also captures the subsequent lock-up as the solid-like material resists further deformation. A parameter set for 15 lb/ft^3 polyurethane foam is listed in Appendix B of Neilsen et al. (2018). This parameter set was converted to metric units and used for the fiberboard herein with one modification: Poisson's ratio ν was set to zero to avoid lateral expansion in uniaxial compression. Note that it is not immediately obvious from Section 5 of Neilsen et al. (2018) that setting $\nu = 0$ eliminates lateral expansion, but the lateral expansion was confirmed to be two orders of magnitude less than the axial strain magnitude in a uniaxial compression simulation, as desired. No failure criterion was specified.

The uniaxial compressive behavior of the Flexible Foam model was computed and plotted in Fig. 2-4 as "Sim, Flexible Foam". The model clearly captures the overall shape of the experimental measurements, but it does not match the measurements perfectly. The model parameters and functions could be adjusted to improve the fit to the experimental data, but resources were not devoted to that endeavor. The model is isotropic, so it will never capture the

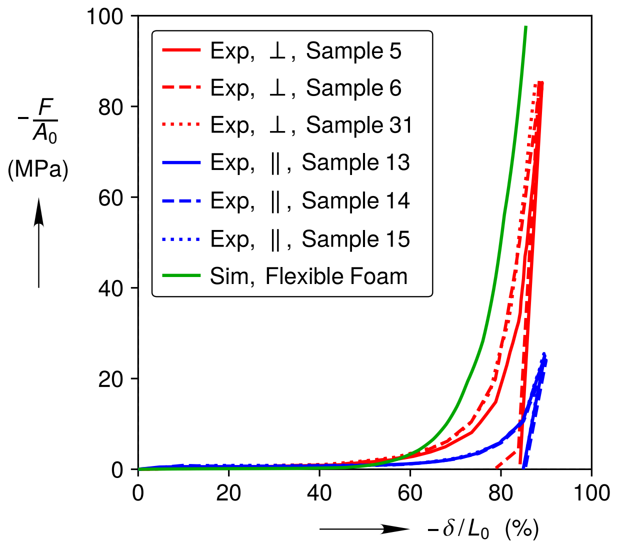


Figure 2-4. Fiberboard compressive engineering stress–strain measurements, perpendicular (⊥) and parallel (||) to the fiber, from Walker (1991), compared against a single element simulation using the Flexible Foam model defined herein.

fiberboard's anisotropy, and it seemed prudent to assess fiberboard's impact on container compaction using this fair approximation of the fiberboard behavior before attempting to improve the fit.

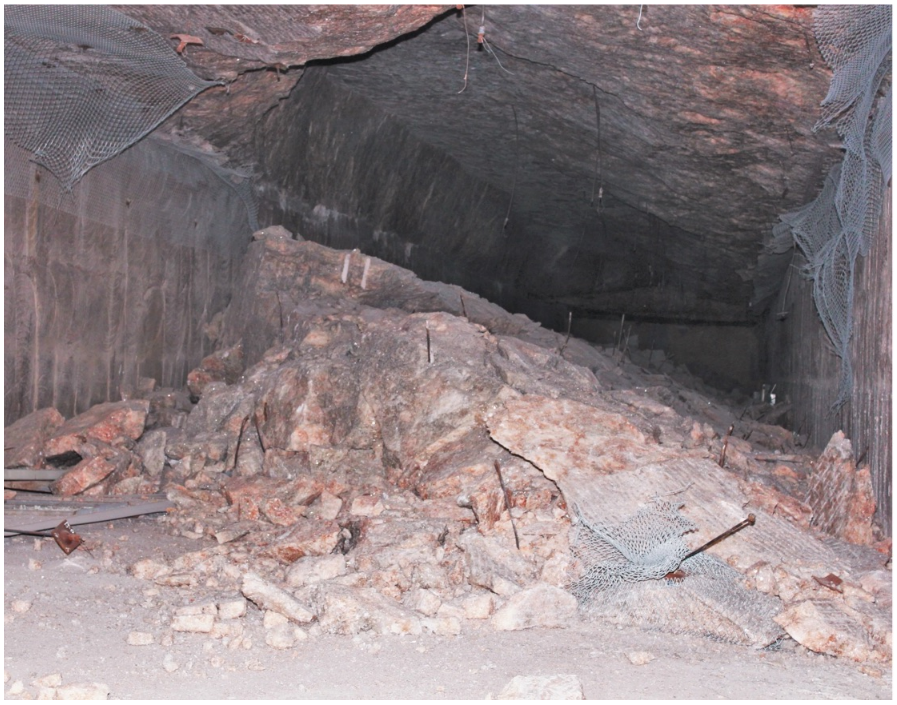
2.2. Roof Fall Compaction Model

Roof falls can come in all shapes and sizes, so determination of the absolute worst-case roof fall scenario for container clustering was deemed too difficult. We also did not attempt to find a statistical worst-case by running hundreds of roof fall simulations because each roof fall simulation required a week or more to run. Instead, a number of conservative assumptions were adopted to make a "severe case", as described in the following subsections.

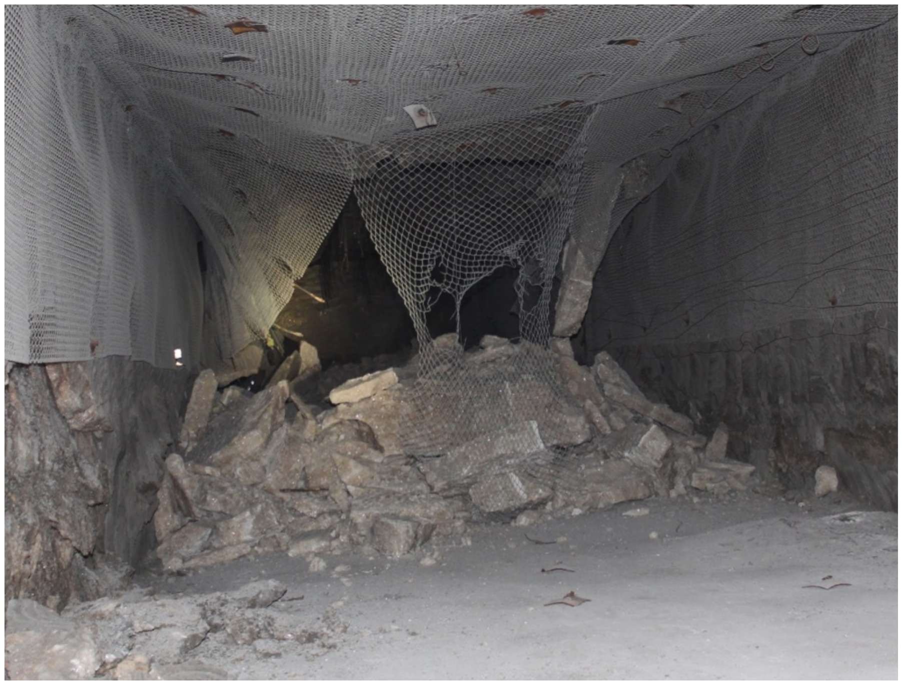
2.2.1. Geomechanical Model

Roof falls are the result of fractures above a room, but fracturing is a difficult processes to simulate in the finite element method. A host of numerical techniques have been developed to alleviate the difficulties over the last several decades, yet these techniques are still topics of research. Constitutive models for rock salt fracturing are also topics of research, and the existing models in the literature have not been implemented into Sierra/Solid Mechanics (2019). For these reasons, the natural fracture process above the room was not simulated. Instead, fractures were specified a-priori based on previous observations of rock falls at the WIPP.

Several roof falls have occurred at the WIPP recently because rock bolting was not possible for several years after the fire and radiological release events in February 2014. Roof falls tend to



(a) Lower horizon (Panel 7, Room 4). (Photograph flipped left-to-right.)



(b) Upper horizon (E300-S3650).

Figure 2-5. Photographs of rubble piles resulting from roof falls at the WIPP (Carrasco 2019b).

detach at anhydrite stringers or at clay seams. Both detachment modes occur no higher than the first clay seam above the room, and the distance from the lower horizon rooms to clay seam G is larger than the distance from the upper horizon rooms to clay seam H. Accordingly, roof falls on the lower horizon tend to involve large blocks of salt (see Fig. 2-5a), while roof falls on the upper horizon tend to involve a number of smaller blocks that detach at anhydrite stringers (see Fig. 2-5b) (Kicker et al. 2017). The larger roof fall blocks on the lower horizon are believed to be more likely to cause container clustering. The thickest block of salt to fall at WIPP occurred on the lower horizon when a block detached at clay seam G in Panel 7, Room 4 (see Fig. 2-5a) (Carrasco 2019a). A thicker slab of salt than Panel 7, Room 4, is unlikely because a thicker slab would have to somehow remain intact across clay seam G.

The estimated dimensions of the Panel 7, Room 4, roof fall (Carrasco 2019a) were utilized to a-priori specify the trapezoidal shaped roof fall geometry shown in “Roof Fall Compaction Setup” in Fig. 2-1. The trapezoid was $L_G = 2.43$ m thick on the left side and 0.07 m thick on the right side. The weight imbalance of this geometry has the potential to crush the leftmost POP containers and/or push the POP containers into a cluster on the right side of the room. Although the actual roof fall in Fig. 2-5a spanned the width of the room, 0.25 m was removed off from each end to ensure the free falling block was not impeded by the walls of the room. This trimming resulted in a block width of $L_b = 9.56$ m at the bottom and $L_t = 2.19$ m at the top. See Appendix A.4.1 for the Cubit journal file used to create the finite element mesh, and see Appendix A.3 for an image and description of the rock fall mesh.

The trapezoidal roof fall block was tied (bonded) to the surrounding rock mass during the fluid pressure reduction phase (instantaneous room excavation phase) described at the end of Section 2.1.1. At $t = 0$, the roof fall block completely detached from the surrounding rock mass so that it freely fell on top of the containers. In reality, rock bolts typically secure the damaged roof to the intact surrounding rock mass, which can prevent roof falls for decades, provided the rock bolts are regularly replaced. By dropping the rock fall block at $t = 0$ in the simulation, the gap between the roof and the container was not allowed to decrease due to room closure.

By detaching the block from all the surrounding rock mass simultaneously, the block was made to land (nearly) flat on the POP containers herein. If, instead, the block was progressively detached from the surrounding rock mass, it could rotate about the Y-axis before impacting the containers. For the sake of discussion, assume the block rotated clockwise about the Y-axis. A tilted block could apply lateral loads (in the negative X-direction) to the rightmost containers, potentially causing them to collect on the left side of the room. If the rightmost containers impacted by a tilted block supply sufficient resistance, however, they could apply a moment to the block, causing it to rotate counterclockwise about the Y-axis. This rotation reversal could cause the block to land flat on the containers, or it could be large enough to make the left side of the block impact the containers on the left side of the room. In any case, the total kinetic energy imparted to the containers would be the same, only the spatial and temporal distribution of the energy transfer would change. Rotating blocks were not investigated herein, but may be worthwhile to study in future analyses.

If a salt block was to land on POP containers, the impact could fracture the block into smaller pieces or cause microfractures inside the block. Fracturing dissipates energy due to the creation of new surfaces. In the simulation, however, the salt block was not allowed to fracture upon impacting the POP containers, so all the energy of impact was absorbed by the POP container deformation and salt viscoplasticity.

Given that the block cannot break into smaller pieces in our simulations, one might ask, “Could small or medium size blocks on top of the containers eventually get wedged in-between containers during the compaction process?” To consider this question, blocks smaller than a POP container will be referred to as small blocks, while blocks somewhere between the size of a POP container and the large block in Fig. 2-1 will be referred to as medium blocks. Presuming the large block of salt in Fig. 2-1 fails to substantially deform the containers, then a roof fall of small or medium sized blocks would simply settle on top of the containers. A roof fall creates a cavity above the rubble pile sitting on top of the containers, but the surrounding rock mass will eventually close in and contact the rubble pile. A rubble pile of small blocks would probably just rearrange to mate up with with the roof before transferring any substantial load to the containers. A well distributed load on top of the containers would be similar to the gradual compaction simulations. Some smaller blocks would find their way in-between the containers, but they would likely only fill the interstitial sites between the containers. This would make the average spacing between containers increase since there would be more material to resist the lithostatic stresses applied by the surrounding rock mass. If medium sized blocks, on the other hand, were to fall on the containers, the roof fall would probably produce a cavity above the blocks that is similar in shape to the blocks. After creeping inward, the surrounding rock mass should mate up somewhat with the medium size blocks, meaning excessive pressure probably will not be applied to any one block to drive it in-between the containers. For these reasons, we assumed small and medium sized blocks were less of a concern than a large roof fall block, and further analysis of small and medium blocks was left as potential future work.

2.2.2. Container Model

The container model for the roof fall compaction simulation was similar to the gradual compaction container model, except for two notable differences:

1. Slightly weaker hardening curves were utilized for the stainless and carbon steel to conservatively encourage them to deform more during the roof fall. (See Appendix A.2 for details of the weak hardening curves.) The plywood and fiberboard dunnage, on the other hand, were not weakened. The plywood strength had already been reduced to 20 % of the short-term measured value to capture wood failure’s dependence on the loading duration, as discussed in Section 2.2.2.3 of Reedlunn et al. (2019). The fiberboard was considered sufficiently compliant prior to lock-up to be negligible.

2. The weight imbalance of the roof fall in Fig. 2-1 has the potential to push containers towards the right side of the room (positive *X* direction). As such, each POP seven-pack was stacked 25.4 mm to the right of the 7-pack beneath it to further encourage the POP stacks to topple rightwards in the roof fall simulation. Otherwise, the containers were emplaced in the same manner described in Section 2.2.1 of Reedlunn et al. (2019) and shown in Fig. 2-3. See Appendix A.4.1 for Cubit journal files used to emplace the containers.

In addition to the aforementioned differences, two remaining aspects of the container model are worth calling attention to:

1. The MgO sacks were again conservatively omitted from the roof fall simulation. Without MgO sacks to cushion the fall, the block fell further and gathered more momentum before impacting the containers.

2. Stainless steel, carbon steel, plywood, and fiberboard all exhibit higher strengths at higher strain rates. To be conservative, the stainless steel, carbon steel, and plywood material models did not include this effect: the models were rate independent and calibrated against slow strain rate experiments. The fiberboard model was rate dependent, but the fiberboard should have provided very little resistance to deformation prior to lock-up, even at faster strain rates (see Figure 21 in Neilsen et al. (2018)).

2.3. Numerical Approach

The severe deformations, material failure, and pervasive contact in these simulations made them challenging, if not nearly impossible, to run in a finite element code that implicitly integrates the momentum balance equations. Reedlunn et al. (2019), however, sped up the salt's viscoplasticity to squeeze 1000 years into a few seconds so that Sierra/Solid Mechanics (2019) could explicitly integrate the momentum balance equations. The gradual compaction simulations herein utilized the same viscoplastic rate scaling profile selected in Section A.2.1 of Reedlunn et al. (2019). The roof fall compaction simulation, on the other hand, did not employ any viscoplastic rate scaling from $t = 0$ to 2.0 s in order to let the roof block fall and settle onto the POP containers without decades of room closure occurring during the same time period. Following the roof fall, the salt's viscoplasticity was sped up in attempts to simulate the subsequent gradual compaction, but such attempts were unsuccessful. The thin right end of the salt block on top of the containers would become heavily distorted and either the material model would fail or elements would invert. We tried to avoid these failures by killing elements with an equivalent stress above 50 MPa, but the material model still failed to converge prior to reaching 50 MPa. Further troubleshooting could be fruitful, but was left for potential future work. In the mean time, the gradual compaction simulations should serve as a reasonable estimate of the final compacted configuration after a roof fall.

As is commonly done, elements were deleted from the simulation when container elements were heavily distorted or when container materials reached their failure criterion. No elements were deleted from the roof fall simulations, while Table 2-1 lists the percentages of container elements deleted by the end of the gradual compaction simulations, grouped by material. In reality, of course, material does not disappear when it severely deforms or fractures. Although numerical techniques exist that allow materials to deform and fracture without deleting elements, such techniques are research subjects and left for potential future work. The substantial loss of the carbon steel, plywood, and fiberboard elements in the gradual compaction simulations should conservatively increase compaction, but not excessively, as the stainless steel pipes largely persist and provide the bulk of the compaction resistance.

Table 2-1. Percentages of container elements deleted after $t = 1000$ yr of gradual compaction.

Case	Carbon Steel (%)	Plywood (%)	Fiberboard (%)	Stainless steel (%)
6-inch Standard	56.6	100	97.2	0.68
12-inch Standard	30.6	100	89.1	2.6

Reedlunn et al. (2019) scaled the mass densities of smaller finite elements in the containers to meet a target momentum balance (explicitly integrated) time step of 4.0 μs. Herein, the same mass density scaling was utilized in the gradual compaction simulations, but not in the roof fall compaction simulation. Dynamic events ensued when the roof fall landed on the containers, so it was important to avoid artificially adjusting the container masses.

Sierra/Solid Mechanics input files for both the gradual compaction and roof fall simulations can be found in Appendix A.4.2.

3. RESULTS

This section presents the roof fall compaction results (Section 3.1) before the gradual compaction results (Section 3.2) since roof falls occur prior to gradual compaction in reality.

3.1. Roof Fall Compaction

Images of the roof fall compaction simulations at selected time instances are shown in Fig. 3-1. Although the containers and geology are displayed normal to the *X*-*Z* plane, the simulations were, in fact, three-dimensional. Note that the plywood, lateral fiberboard, and 55-gallon drums were included in both simulations, but hidden from view in Fig. 3-1 to reveal the flanged pipe deformation during the roof fall impact. The top and bottom fiberboard are shown in red, the plywood is brown, the flanged pipe is blue, and the roof fall block is shown in dark grey to distinguish it from the surrounding rock mass in light grey. Both simulations produced similar results, so they will be described together. The block begins its free fall at $t = 0$ s and first contacts the containers at $t = 0.55$ s. Close inspection reveals that the right, thinner, side of the block has contacted the rightmost POP containers, while a gap exists between the left, thicker, side of the block and the leftmost POP containers at $t = 0.55$ s. This slight asymmetry may be due to some sliding at clay seam G and/or sagging of the (new created) roof as the block was released. By $t = 0.60$ s, the block has rotated about the *Y*-axis to compress the leftmost containers. The impact then causes the right side of the block to tilt upwards and the top container layer to bounce. A gap is clearly visible between the top and middle container layers at $t = 0.75$ s, but the block settles on top of the containers and everything becomes relatively stationary between $t = 0.75$ s and 2.00 s. The containers are not stacked quite as neatly at $t = 2.00$ s as they were upon emplacement. For example, a 6-inch Standard container at the top right corner at $t = 2.00$ s appears close to tipping over into the gap between the containers and the right wall.

On the whole, the roof fall simulations in Fig. 3-1 predict almost negligible container deformation and container rearrangement from the roof fall. Although some of the fiberboard is mildly distorted and the 55-gallon drums on the left side of the room exhibit dents (not shown in Fig. 3-1), these effects are rather minor. As the room gradually closes over the next 1000 yr, a few small deformations, and potentially one or two tipped over containers, at $t = 2.00$ s are unlikely to significantly change the final compacted geometry.

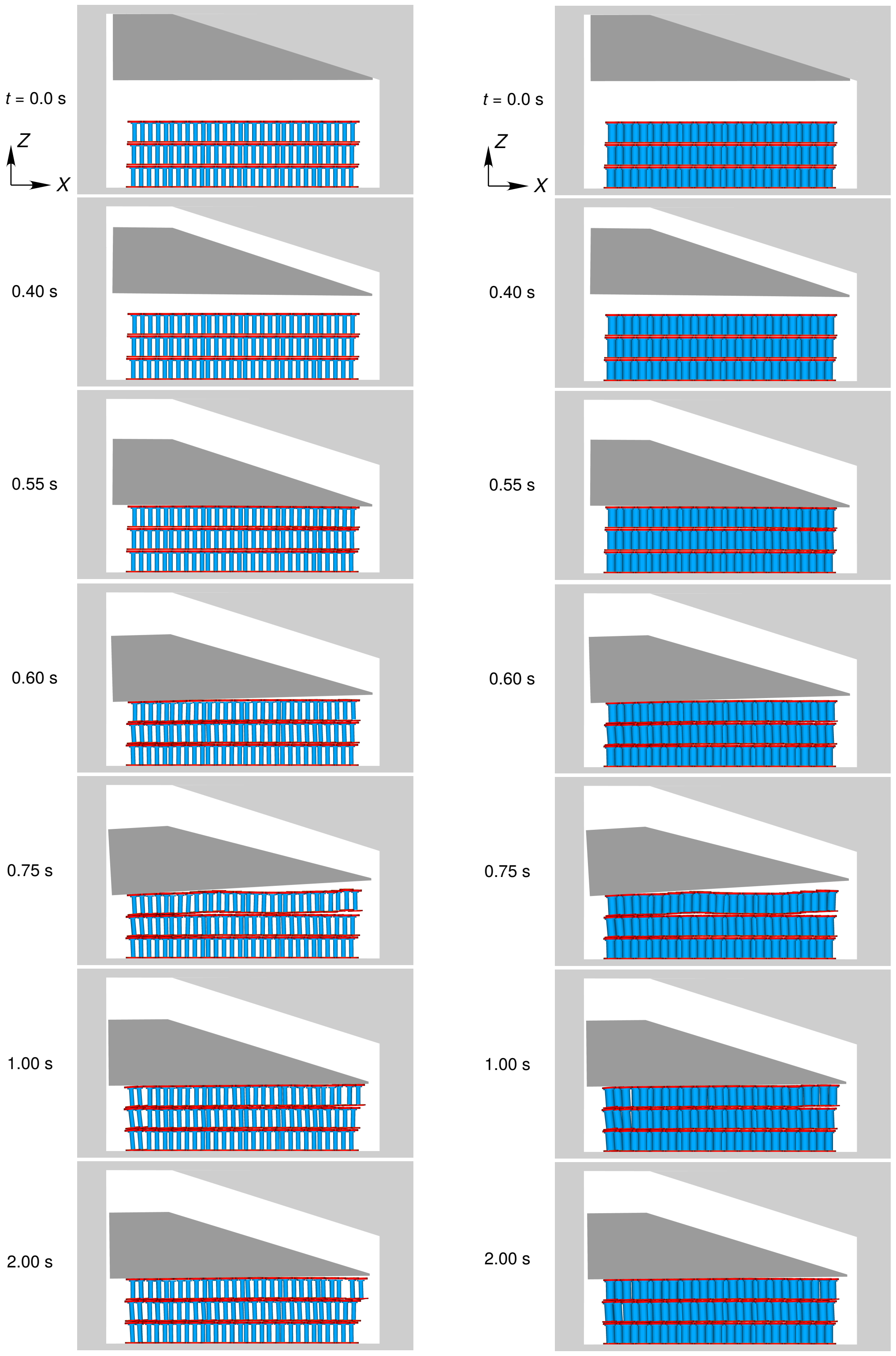
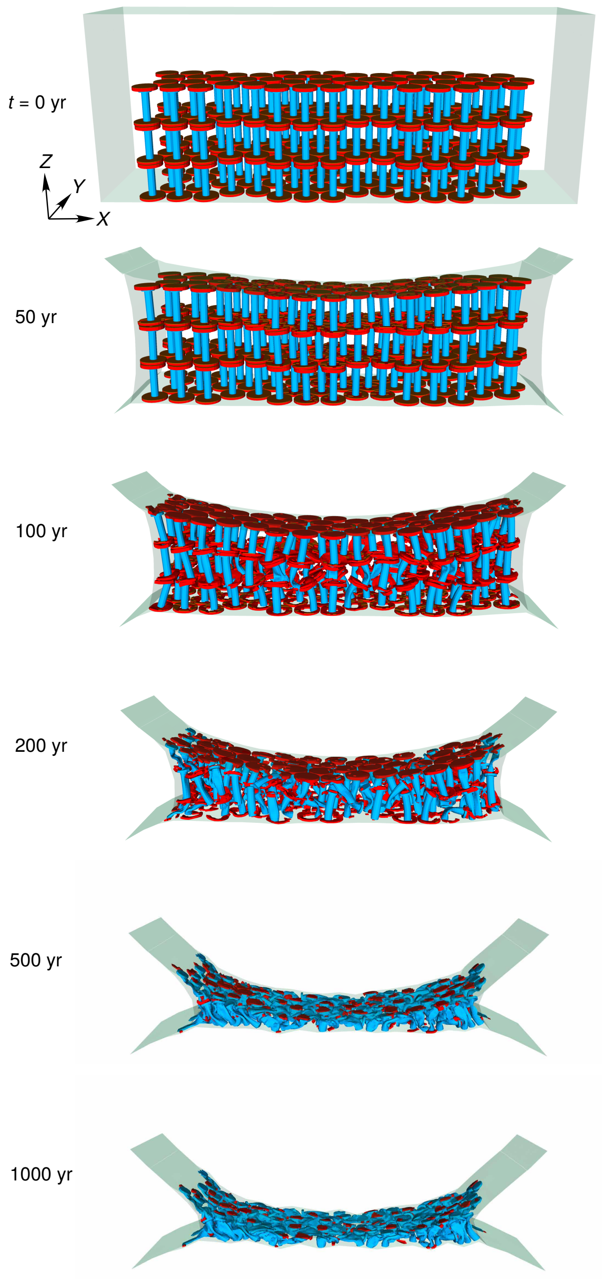
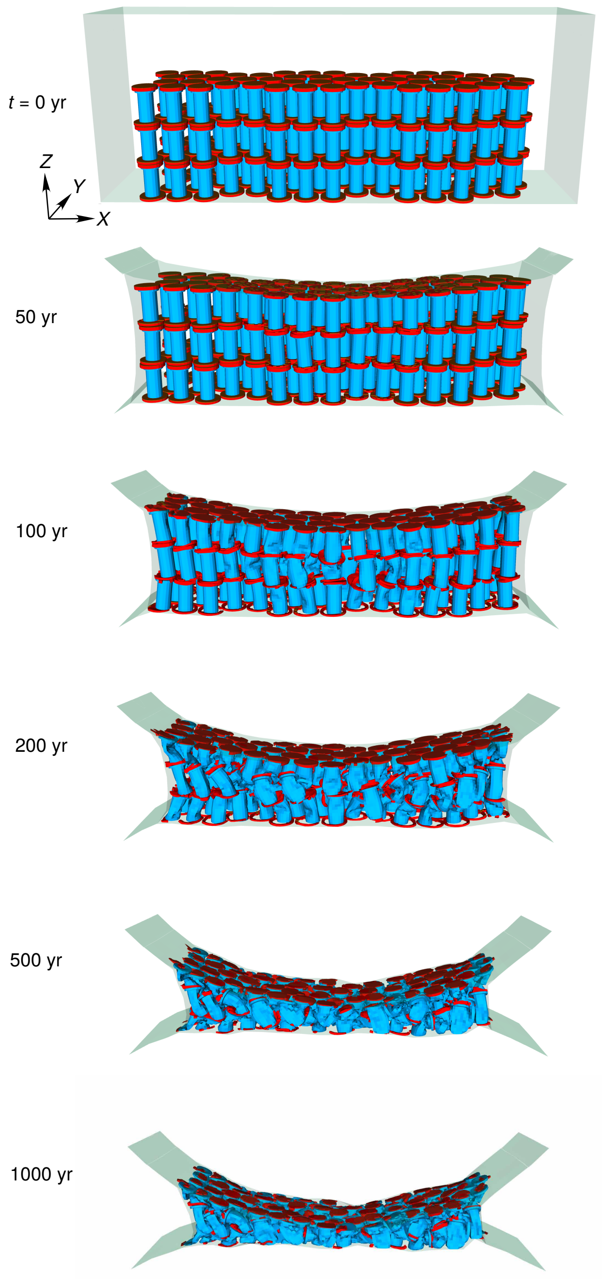


Figure 3-1. Images of a roof fall block impacting 6-inch and 12-inch Standard POP containers at selected instances in time. (Lateral fiberboard and 55-gallon drums hidden for visualization purposes.)

3.2. Gradual Compaction



(a) 6-inch Standard



(b) 12-inch Standard

Figure 3-2. Images of gradually compacted 6-inch and 12-inch Standard POP containers at selected instances in time. (Lateral fiberboard, 55-gallon drums, and geology hidden for visualization purposes.)

As discussed in Section 2.3, the gradual compaction simulations discussed in this section include room closure due to salt viscoplasticity, but do not include roof falls. The final compacted shape of containers after 1000 yr of room closure is presumed to be insensitive to whether the roof fell on top of the containers or not.

Images of 6-inch and 12-inch Standard container gradual compaction predictions are shown in Fig. 3-2 at selected time instances. As before, the plywood, lateral fiberboard, and 55-gallon drums were included in the simulation, but are hidden in Fig. 3-2 to reveal the flanged pipes. The geology is also hidden, leaving only the semi-transparent surface of the room, and the perspective view is tilted, in order to show the three-dimensional nature of the compaction. The roof first contacts the containers at room mid-width at $t = 41$ yr and contacts the entire upper layer of containers by $t = 70$ yr. By $t = 100$ yr, many 6-inch and 12-inch flanged pipes have buckled under the axial load supplied by the roof and floor. By $t = 200$ yr, numerous pipes originally stacked in the upper and middle container layers have been pushed down into the bottom layer. Further room closure buckles the remaining flanged pipes and further pushes the upper/middle container layers down into the bottom container layer. At $t = 1000$ yr, the compacted shapes of both 6-inch and 12-inch Standard container arrays are essentially stabilized to something resembling a tall and slender "I" turned on its side, but the 6-inch Standard containers are noticeably more compacted.

As discussed in Chapter 1, Park and Hansen (2004) and Park and Hansen (2005) previously predicted the compaction of 6-inch and 12-inch Standard POP containers using a homogenized approach that was far less computationally intensive than the discrete approach employed herein, so a comparison is enlightening. The difference between Fig. 1-2 and Fig. 3-2 at $t = 1000$ yr is dramatic. Park and Hansen (2005) predicted roughly 13 % and 9 % reductions in 6-inch and 12-inch Standard container stack heights at the room mid-width, respectively, while our simulations predict 93 % and 82 % reductions in 6-inch and 12-inch Standard container stack heights at the room mid-width, respectively. The shape of the waste container outer envelope differs as well. Park and Hansen (2005) predicted the most vertical compaction at the left and right flanks of the waste, while our simulations predict the most vertical compaction at the center of the room. Although a large portion of these differences is due to our discrete approach, as opposed to their homogenized approach, some portion may be due to differences in the material models for the POP containers. Although a direct comparison does not exist at this juncture, the discrete approach presented herein clearly predicts more compaction and is hence conservative for supporting nuclear criticality assessments.

The images in Fig. 3-2 provide a qualitative view of how the containers deform, but the closure histories in Fig. 3-3 give a more quantitative viewpoint. The horizontal and vertical closures, δ_h and δ_v, have been respectively normalized by L_h and L_v in Fig. 3-3 to make them into horizontal and vertical closure percentages. The empty room closure curves represent the possibility that all canister materials degrade to zero structural stiffness. In the absence of containers to restrain the empty room closure, the vertical closure, $\delta_\mathrm{v}/L_\mathrm{v}$ quickly reaches 100 % by $t = 180$ yr. As one might expect, the empty room $\delta_\mathrm{h}/L_\mathrm{h}$ curve exhibits a mild kink once the roof touches the floor and $\delta_\mathrm{h}/L_\mathrm{h}$ becomes virtually flat at $t \approx 400$ yr when the empty room completely closes. On the other hand, the closure curves for the two POP compaction simulations only match the empty room closure curves until $t = 41$ yr. At $t = 41$ yr, the roof contacts the containers and the POP room closure curves start to diverge from the empty room closure curves. The POP room closure curves diverge more significantly at 63 yr when the roof more fully contacts the containers. Beyond $t = 63$ yr, the POP compaction closure curves gradually asymptote to their stable final values. The room filled with 6-inch Standard containers closes more than that filled with 12-inch Standard containers, as expected, but it also horizontally closes slightly more than the empty

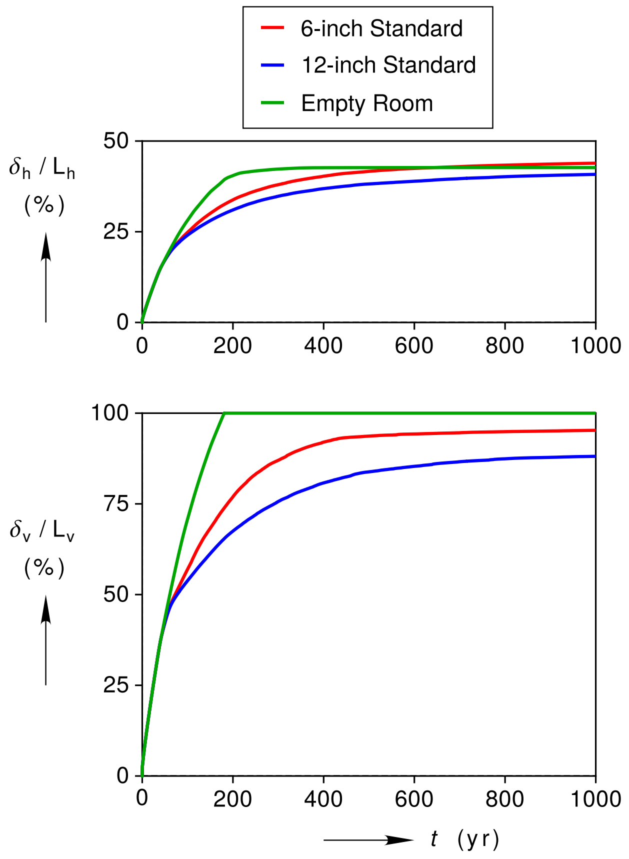


Figure 3-3. Gradual compaction horizontal and vertical closure histories.

room. The horizontal closure is less for the empty room because it closes vertically more quickly, which pinches off the horizontal closure more quickly than the room filled with 6-inch Standard containers. The closure percentages at t = 1000 yr are listed in Table 3-1 for reference.

Table 3-1. Closure predictions at $t = 1000$ yr.

Case	δ_h/L_h (%)	δ_v/L_v (%)
6-inch Standard	43.9	95.3
12-inch Standard	40.8	88.1
Empty Room	42.7	100

The gradual compaction results were futher probed by tracking the centroids of each flanged pipe. Instead of calculating the true centroid of every pipe, which would have been challenging once the pipes significantly deformed, the pipe centroid was approximated by averaging the pipe wall nodal coordinates. To compare, the true centroid in the undeformed configuration was 0.325 m from the bottom of the flanged pipe, while averaging the pipe wall nodal coordinates approximated the undeformed centroid as 0.346 m from the flanged pipe bottom. Although this 6 % discrepancy was not ideal, it was considered small enough to be acceptable in the face of more complicated centroid calculation schemes. Each approximate flanged pipe centroid was tracked throughout the gradual compaction simulations. The coordinates at t = 0 and 1000 yr are listed in Appendix B for both POP container compaction simulations.

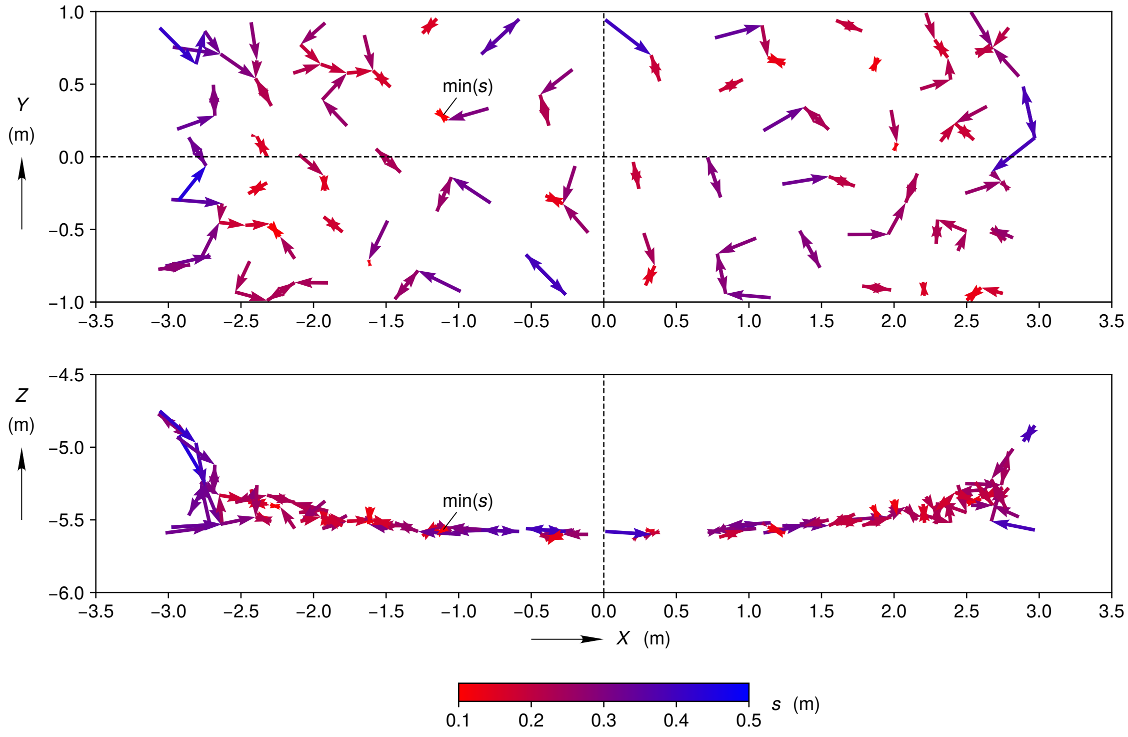
The coordinates in Appendix B were analyzed to find the vector $\boldsymbol{r}$ from each centroid to its nearest neighbor centroid at t = 1000 yr. These vectors are plotted in Fig. 3-4, where the vectors are colored by their Euclidean length s. Significant scatter exists in the results, but some observations can be made. First, the vectors near the sides of the room tend to lie closer to the X-Z plane, while more vectors near the center of the room lie close to the X-Y plane. Second, 6-inch Standard centroids with $s < 0.25$ m tend to occur within $-2.7 < X < 2.7$ m, and the density of small spacings appears highest within $1.5 < |X| < 2.7$ m. Third, the 12-inch Standard centroids are spaced significantly further apart, and the few centroids with $s < 0.25$ m appear to occur within $-2.0 < X < 2.0$ m. These observations are consistent with the t = 1000 yr deformed container images in Fig. 3-2, where the roof and floor sandwich the containers near the center of the room, while the bowed room sides partially shield the containers from the roof-to-floor compaction near the room sides.

Histograms of the distance s are plotted in Fig. 3-5, where N is the number of centroid spacings whose distances fall into a given bin. Each distribution appears close to symmetric, although the right side tail is a little longer than the left in each case. The 6-inch Standard centroid spacings are smaller than the 12-inch Standard centroid spacing, as expected from Figs. 3-2 to 3-4, but the 6-inch Standard centroid spacing also have a wider distribution than the 12-inch centroid spacings.

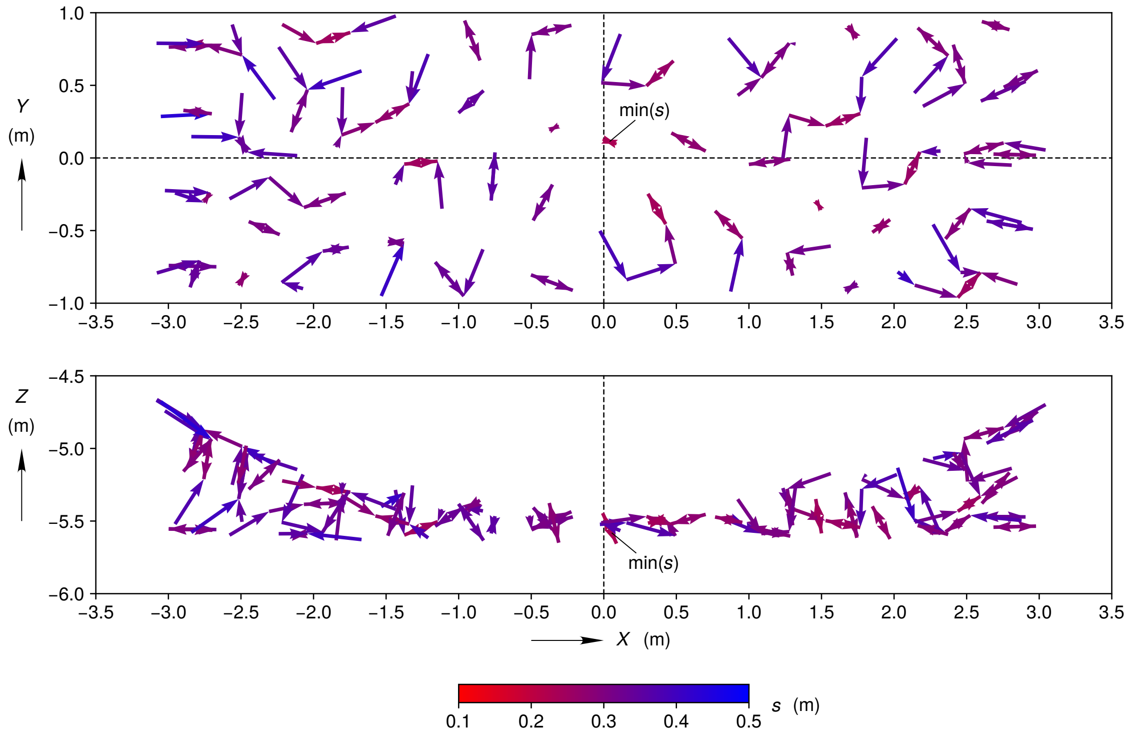
To conclude this section, the minimum, mean, and maximum distance between centroid nearest neighbors are listed in Table 3-2 for reference, along with the centroid coordinates that correspond to the minimum distance s for a given simulation. The $\boldsymbol{r}$ vectors corresponding to the minimum s are also labeled in Fig. 3-4.

Table 3-2. Nearest neighbor statistics at t = 1000 yr.

Case	min(s)	mean(s)	max(s)	Centroids with min(s)		
				X	Y	Z
	(m)	(m)	(m)	(m)	(m)	(m)
6-inch Standard	0.109	0.237	0.442	-1.150	0.318	-5.600
				-1.080	0.251	-5.550
12-inch Standard	0.233	0.315	0.436	0.088	0.096	-5.655
				-0.008	0.128	-5.446

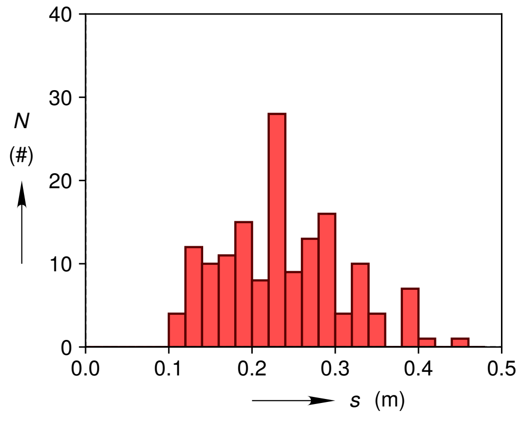


(a) 6-inch Standard

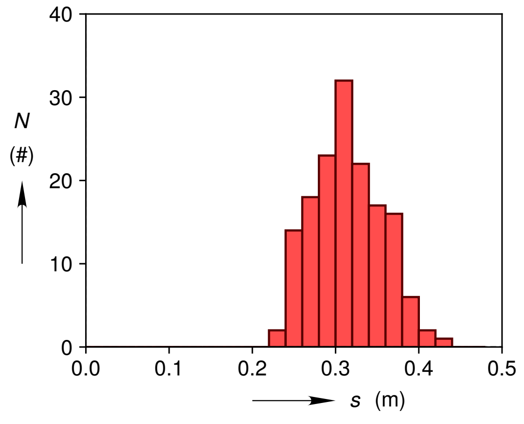


(b) 12-inch Standard

Figure 3-4. Vectors of pipe centroid to nearest neighbor pipe centroid at $t = 1000$ yr from gradual compaction simulations. The vectors are colored according to their Euclidean length s.



(a) **6-inch Standard**



(b) **12-inch Standard**

Figure 3-5. Histograms of pipe centroid to nearest neighbor pipe centroid distances at $t = 1000$ yr from gradual compaction simulations.

4. SUMMARY

This report documents mechanical compaction predictions of 6-inch and 12-inch Standard POP containers due to disposal room closure at the WIPP. These compaction predictions are intended for use in nuclear criticality evaluations performed by ORNL. Following Reedlunn et al. (2019), the plywood dunnage, fiberboard dunnage, and stainless steel flanged pipe inside the 55-gallon carbon steel drum were all discretely modeled to capture the complex mechanical behavior of a single container. A collection of 153 containers within a portion of a disposal room were also discretely modeled to capture how containers interact and slide past one another as they deform. Advancing beyond the analysis approach in Reedlunn et al. (2019), a severe roof fall scenario was simulated herein to evaluate whether a large salt block landing on top of the containers could cause containers to cluster in one part of the room.

Although the simulations utilized a detailed model, it is important to emphasize that a number of uncertainties have not been quantified and the simulations predict centuries into the future with a partially validated model. A number of potential improvements were identified:

1. Geomechanical Model
   a) A roof fall could create a rubble pile of polydispersed blocks on top of the containers. Analyze how such a rubble pile will affect how the containers compact.
   b) Compare container compaction on lower horizon panels vs. upper horizon panels.
   c) Improve computational efficiency by artificially enforcing left-right symmetry with a mirror boundary condition.
   d) Vary uncertain parameters in the geologic material models.
   e) Rotate the falling block about the Y-axis to cause the left (or right) side to impact the containers prior to the right (or left) side.
2. Container Model
   a) Include gas pressure due to container and waste degradation.
   b) Vary or measure the friction coefficients.
   c) Add MgO sacks and/or granules.
   d) Build a homogenized container model to compare against.
   e) Add time-dependent behavior (creep) to the container material models.
   f) Allow container materials to corrode and rot over time.
   g) Model more rows of containers.

h) Perturb the initial container locations.

i) Increase the spacing between the containers to include the drum ribs and bolt ring.

j) Add waste to each container.

3. Numerical

   a) Troubleshoot the numerical issues preventing gradual compaction after a roof fall.

   b) Investigate if adding a small amount of plasticity to plywood behavior allows the plywood to persist further into the simulations.

   c) Model the plywood with shell elements instead of solid elements.

   d) Investigate methods to avoid simply deleting fractured and inverted elements.

   e) Troubleshoot why half-containers cause numerical issues. If resolved, then attempt to use periodic boundary conditions on half-container sliced faces.

   f) Confirm that mass density scaling does not impact the gradual compaction results.

   g) Perform a finite element mesh size sensitivity study.

In light of these potential improvements and uncertainties, the severe roof fall model employed the following conservatisms in an attempt to bound some of the variability in roof falls and container behavior:

1. The thickest roof fall block known to fall at the WIPP was dropped on the containers.
2. The trapezoidal shaped block was thick on the left side and thin on the right side in order to encourage containers to cluster on the right side of the room.
3. Each container layer was stacked 25.4 mm to the right of the container layer beneath it to further encourage clustering on the right side of the room.
4. The block was dropped immediately after room excavation, such that room closure had not diminished the gap between the roof and the containers.
5. The magnesium oxide sacks normally placed on top of every other container stack were omitted to avoid cushioning the salt block as it landed.
6. The block was not allowed to break into smaller pieces, which caused the containers and the salt viscoplasticity to absorb all the salt block's kinetic energy.
7. The increase in stainless steel, carbon steel, and plywood strength at higher strain rates was ignored. (The fiberboard was considered negligible prior to lock-up.)
8. The carbon steel and stainless steel were assigned their minimum ASTM yield strengths and the plywood strength was reduced by 80 % to encourage greater container deformation.

The gradual compaction model also utilized conservatisms to bound some of the variability. The conservative assumptions were:

1. The structural resistance of the waste inside the stainless steel pipes was not included.

2. The magnesium oxide sacks were again omitted.
3. Gas pressure inside the room due to cellulose degradation and metal corrosion was not included.
4. The plywood was assigned a weak failure strength as discussed in Section 2.2.2.3 of Reedlunn et al. (2019).
5. Container elements were deleted from the simulation when an element became severely distorted or a material fractured.

With these caveats and conservatisms in mind, the results presented herein are reasonable predictions of POP container compaction due to room closure. In the roof fall simulation, the block simply settled on top of the containers and caused almost negligible container deformation or scattering. This prediction suggests that a large roof fall landing flat on top of the containers is unlikely to cause container clustering. In the gradual compaction simulations, substantial container buckling occured, with the upper and middle container layers getting partially thrust down into the bottom container layer. The 6-inch Standard containers had minimum and mean pipe centroid spacings of 0.109 m and 0.237 m, respectively, while the 12-inch Standard containers had minimum and mean pipe centroid spacings of 0.233 m and 0.315 m, respectively.

REFERENCES

ASTM (2018a). *Standard Specification for Seamless, Welded, and Heavily Cold Worked Austenitic Stainless Steel Pipes*. Standard A312/A312M-18a. American Society of Testing Materials.

ASTM (2018b). *Standard Specification for Steel, Sheet, Cold-Rolled, Carbon, Structural, High-Strength Low-Alloy, High-Strength Low-Alloy with Improved Formability, Solution Hardened, and Bake Hardenable*. Standard A1008/A1008M-18. American Society of Testing Materials.

Carrasco, R. (Mar. 2019a). *Panel 7, Room 4, Roof Fall Dimensions*. Personal Communication.

Carrasco, R. (Sept. 2019b). *Roof Fall Photographs*. Personal Communication.

Cubit Team (2017). *Cubit mesh generation environment. Users manual.* 15.3. Sandia National Laboratories. Albuquerque, NM, USA; Livermore, CA, USA.

Day, B. (July 2019). *12-inch versus 6-inch Pipe Overpack Containers*. Personal Communication.

Kicker, D., Reedlunn, B., and Herrick, C. (2017). *Analysis Plan for Reconsideration of the WIPP Geomechanical Model for Room Closure*. Tech. rep. AP-178 Rev. 0. September: Sandia National Laboratories.

Kirkes, G. R. (June 2019). *POC data as of June 1 2019*. Archived at WIPP Records Center. ERMS 571584.

Ludwigsen, J., Ammerman, D., and Radloff, H. (Apr. 1998). *Analysis in support of storage of residues in the pipe overpack container*. Tech. rep. SAND98-1003. Albuquerque, NM (United States): Sandia National Laboratories.

McGonagill, S. and Nevarez, J. (2018a). *12 in. Pipe Component*. Tech. rep. POC-DWG-0006, Revision 2. Nuclear Waste Partnership LLC, Carlsbad, NM.

McGonagill, S. and Nevarez, J. (2018b). *12 in. Pipe Component Dunnage*. Tech. rep. POC-DWG-0007, Revision 2. Nuclear Waste Partnership LLC, Carlsbad, NM.

McGonagill, S. and Nevarez, J. (2018c). *12 in. Standard Pipe Overpack*. Tech. rep. POC-DWG-0005, Revision 2. Nuclear Waste Partnership LLC, Carlsbad, NM.

McGonagill, S. and Nevarez, J. (2018d). *POC Rigid Drum Liner*. Tech. rep. POC-DWG-0011, Revision 1. Nuclear Waste Partnership LLC, Carlsbad, NM.

Neilsen, M. K., Lu, W.-Y., Werner, B. T., Scherzinger, W. M., and Lo, C. S. (Mar. 2018). *Flexible Foam Model*. Tech. rep. SAND2018-2433. Sandia National Laboratories.

Park, B. Y. and Hansen, F. D. (2004). *Simulations of the pipe overpack to compute constitutive model parameters for use in WIPP room closure calculations.* Tech. rep. SAND2004-1390. Sandia National Laboratories.

Park, B. Y. and Hansen, F. D. (2005). *Determination of the Porosity Surfaces of the Disposal Room Containing Various Waste Inventories for WIPP PA*. Tech. rep. SAND2005-4236. Sandia National Laboratories (SNL-NM).

Park, B. Y. and Holland, J. F. (2007). *Structural evaluation of WIPP disposal room raised to Clay Seam G*. Tech. rep. SAND2007-3334. Sandia National Laboratories (SNL-NM).

Porter, S. (2013). *Pipe Overpack Type A Evaluation Report*. Tech. rep. POC-REP-0001, Rev. 1. Nuclear Waste Partnership LLC.

Rechard, R. P., Sanchez, L. C., Stockman, C. T., and Trellue, H. R. (2000). *Consideration of nuclear criticality when disposing of transuranic waste at the Waste Isolation Pilot Plant*. Tech. rep. SAND99-2898. Sandia National Laboratories.

Reedlunn, B., Bean, J., Wilkes, J., and Bignell, J. (2019). *Simulations of Criticality Control Overpack Container Compaction at the Waste Isolation Pilot Plant*. Tech. rep. SAND2019-3106 O. Sandia National Laboratories.

Saylor, E. M. and Scaglione, J. M. (Nov. 2018). *Nuclear Criticality Safety Assessment of Potential Plutonium Disposition at the Waste Isolation Pilot Plant*. Tech. rep. ORNL/TM-2017/751/R1. Oak Ridge National Laboratory (ORNL).

Sierra/Solid Mechanics (2018). *Sierra/Solid Mechanics User's Guide*. 4.50. SAND2018-10673. Sandia National Laboratories. Albuquerque, NM, USA; Livermore, CA, USA.

Sierra/Solid Mechanics (2019). *Sierra/Solid Mechanics User's Guide*. 4.52. SAND2019-2715. Sandia National Laboratories. Albuquerque, NM, USA; Livermore, CA, USA.

Skolnik Industries (Nov. 2018). *Carbon Steel Drums*. http://www.skolnik.com/product.php?product=carbon_steel_drums. Accessed: 2018-11-16.

Smith, A. C. and Vormelker, P. R. (2000). *Celotex Structural Properties Tests*. Tech. rep. WSRC-TR-2000-00444. Westinghouse Savannah River Company. DOI: 10.2172/773934.

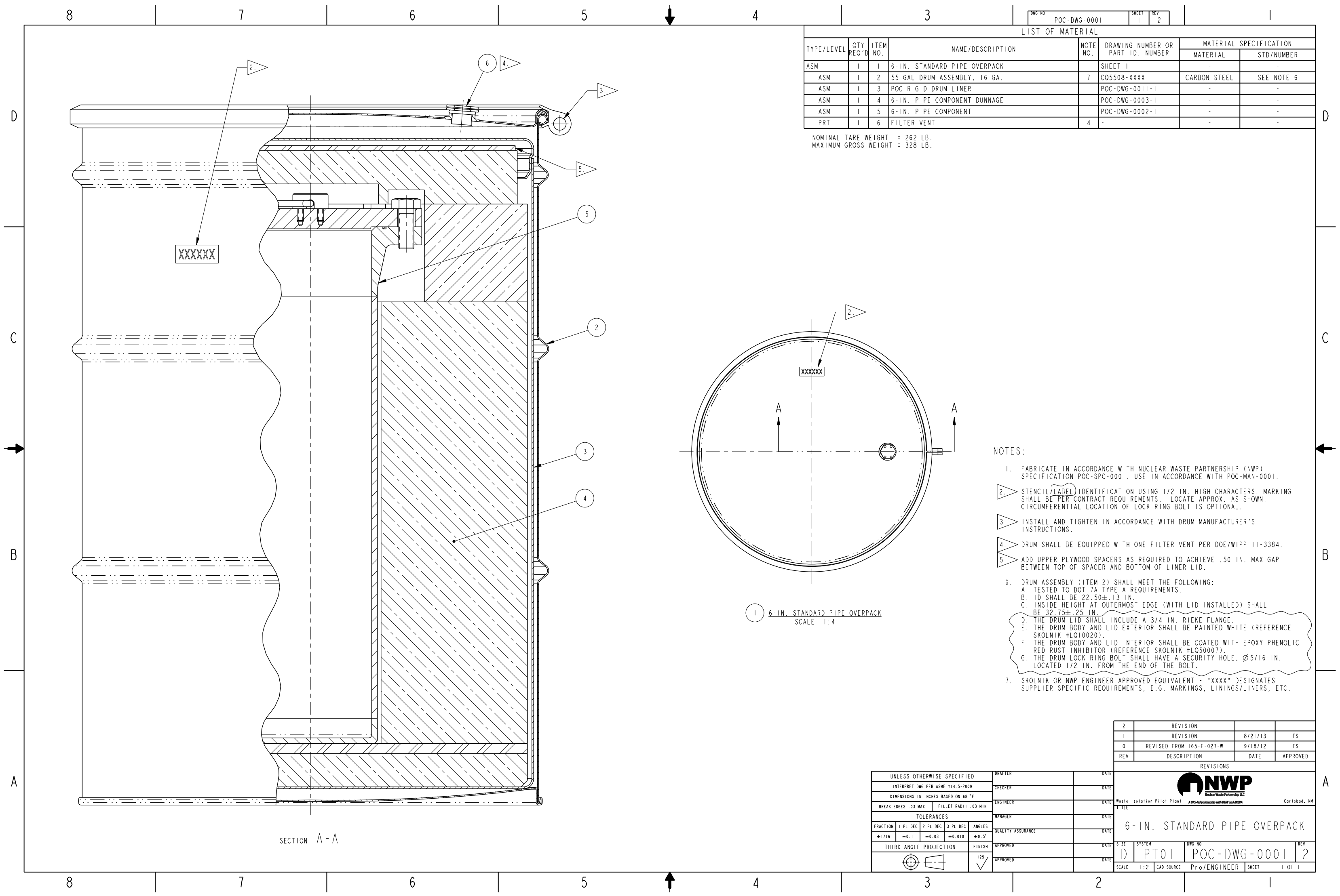
Stone, C. M. (Aug. 1997). *Final disposal room structural response calculations*. Tech. rep. SAND97-0795. Albuquerque, NM, USA; Livermore, CA, USA: Sandia National Laboratories.

US DOE (July 2013). *CCP Transuranic Authorized Methods for Payload Control (CCP CH-TRAMPAC)*. Tech. rep. CCP-PO-003. United States Department of Energy.

Walker, M. (Jan. 1991). *Packaging materials properties data*. Tech. rep. Y/EN-4120. Oak Ridge Y-12 Plant, TN (United States).

APPENDIX A. Model Setup Details

A.1. Container Technical Drawings



DWG NO POC-DWG-0001 SHEET 1 REV 2

LIST OF MATERIAL

TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DRAWING NUMBER OR PART ID. NUMBER	MATERIAL SPECIFICATION	
						MATERIAL	STD/NUMBER
ASM	1	1	6-IN. STANDARD PIPE OVERPACK		SHEET 1	-	-
ASM	1	2	55 GAL DRUM ASSEMBLY, 16 GA.	7	CQ5508-XXXX	CARBON STEEL	SEE NOTE 6
ASM	1	3	POC RIGID DRUM LINER		POC-DWG-0011-1	-	-
ASM	1	4	6-IN. PIPE COMPONENT DUNNAGE		POC-DWG-0003-1	-	-
ASM	1	5	6-IN. PIPE COMPONENT		POC-DWG-0002-1	-	-
PRT	1	6	FILTER VENT	4	-	-	-

NOMINAL TARE WEIGHT = 262 LB.
MAXIMUM GROSS WEIGHT = 328 LB.

NOTES:

1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001. USE IN ACCORDANCE WITH POC-MAN-0001.
2. STENCIL/LABEL IDENTIFICATION USING 1/2 IN. HIGH CHARACTERS. MARKING SHALL BE PER CONTRACT REQUIREMENTS. LOCATE APPROX. AS SHOWN. CIRCUMFERENTIAL LOCATION OF LOCK RING BOLT IS OPTIONAL.
3. INSTALL AND TIGHTEN IN ACCORDANCE WITH DRUM MANUFACTURER'S INSTRUCTIONS.
4. DRUM SHALL BE EQUIPPED WITH ONE FILTER VENT PER DOE/WIPP 11-3384.
5. ADD UPPER PLYWOOD SPACERS AS REQUIRED TO ACHIEVE .50 IN. MAX GAP BETWEEN TOP OF SPACER AND BOTTOM OF LINER LID.
6. DRUM ASSEMBLY (ITEM 2) SHALL MEET THE FOLLOWING:
   A. TESTED TO DOT 7A TYPE A REQUIREMENTS.
   B. ID SHALL BE 22.50±.13 IN.
   C. INSIDE HEIGHT AT OUTERMOST EDGE (WITH LID INSTALLED) SHALL BE 32.75±.25 IN.
   D. THE DRUM LID SHALL INCLUDE A 3/4 IN. RIEKE FLANGE.
   E. THE DRUM BODY AND LID EXTERIOR SHALL BE PAINTED WHITE (REFERENCE SKOLNIK #LQ10020).
   F. THE DRUM BODY AND LID INTERIOR SHALL BE COATED WITH EPOXY PHENOLIC RED RUST INHIBITOR (REFERENCE SKOLNIK #LQ50007).
   G. THE DRUM LOCK RING BOLT SHALL HAVE A SECURITY HOLE, Ø5/16 IN. LOCATED 1/2 IN. FROM THE END OF THE BOLT.
7. SKOLNIK OR NWP ENGINEER APPROVED EQUIVALENT - "XXXX" DESIGNATES SUPPLIER SPECIFIC REQUIREMENTS, E.G. MARKINGS, LININGS/LINERS, ETC.

REV	DESCRIPTION	DATE	APPROVED
2	REVISION		
1	REVISION	8/21/13	TS
0	REVISED FROM 165-F-027-W	9/18/12	TS

REVISIONS

UNLESS OTHERWISE SPECIFIED				
INTERPRET DWG PER ASME Y14.5-2009				
DIMENSIONS IN INCHES BASED ON 68 °F				
BREAK EDGES .03 MAX	FILLET RADII .03 MIN			
TOLERANCES				
FRACTION	1 PL DEC	2 PL DEC	3 PL DEC	ANGLES
±1/16	±0.1	±0.03	±0.010	±0.5°
THIRD ANGLE PROJECTION				FINISH 125

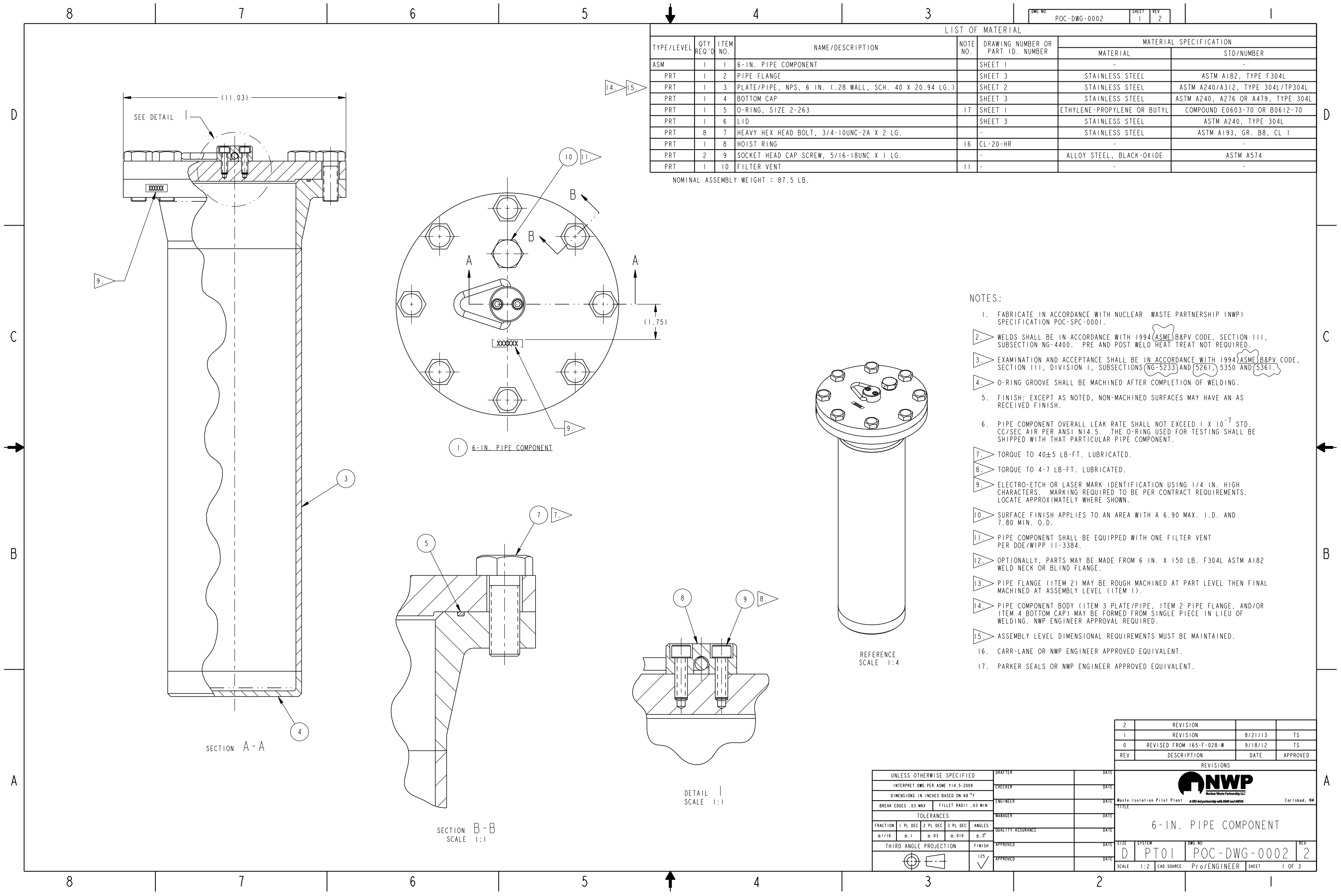
DRAFTER	DATE
CHECKER	DATE
ENGINEER	DATE
MANAGER	DATE
QUALITY ASSURANCE	DATE
APPROVED	DATE
APPROVED	DATE

NWP Nuclear Waste Partnership LLC
Waste Isolation Pilot Plant A URS-led partnership with B&W and AREVA Carlsbad, NM

TITLE 6-IN. STANDARD PIPE OVERPACK

SIZE	SYSTEM	DWG NO	REV
D	PT01	POC-DWG-0001	2
SCALE 1:2	CAD SOURCE Pro/ENGINEER	SHEET 1 OF 1	

Figure A-1. 6-inch Standard Pipe Overpack Assembly Drawing (McGonagill and Nevarez 2018c).



LIST OF MATERIAL							
TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DRAWING NUMBER OR PART ID. NUMBER	MATERIAL SPECIFICATION	
						MATERIAL	STD/NUMBER
ASM	1	1	6-IN. PIPE COMPONENT		SHEET 1	-	-
PRT	1	2	PIPE FLANGE		SHEET 3	STAINLESS STEEL	ASTM A182, TYPE F304L
PRT	1	3	PLATE/PIPE, NPS, 6 IN. (.28 WALL, SCH. 40 X 20.94 LG.)		SHEET 2	STAINLESS STEEL	ASTM A240/A312, TYPE 304L/TP304L
PRT	1	4	BOTTOM CAP		SHEET 3	STAINLESS STEEL	ASTM A240, A276 OR A479, TYPE 304L
PRT	1	5	O-RING, SIZE 2-263	17	SHEET 1	ETHYLENE-PROPYLENE OR BUTYL	COMPOUND E0603-70 OR B0612-70
PRT	1	6	LID		SHEET 3	STAINLESS STEEL	ASTM A240, TYPE 304L
PRT	8	7	HEAVY HEX HEAD BOLT, 3/4-10UNC-2A X 2 LG.		-	STAINLESS STEEL	ASTM A193, GR. B8, CL 1
PRT	1	8	HOIST RING	16	CL-20-HR	-	-
PRT	2	9	SOCKET HEAD CAP SCREW, 5/16-18UNC X 1 LG.		-	ALLOY STEEL, BLACK-OXIDE	ASTM A574
PRT	1	10	FILTER VENT	11	-	-	-

NOMINAL ASSEMBLY WEIGHT = 87.5 LB.

NOTES:

1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001.
2. WELDS SHALL BE IN ACCORDANCE WITH 1994 ASME B&PV CODE, SECTION III, SUBSECTION NG-4400. PRE AND POST WELD HEAT TREAT NOT REQUIRED.
3. EXAMINATION AND ACCEPTANCE SHALL BE IN ACCORDANCE WITH 1994 ASME B&PV CODE, SECTION III, DIVISION 1, SUBSECTIONS NG-5233 AND 5261, 5350 AND 5361.
4. O-RING GROOVE SHALL BE MACHINED AFTER COMPLETION OF WELDING.
5. FINISH: EXCEPT AS NOTED, NON-MACHINED SURFACES MAY HAVE AN AS RECEIVED FINISH.
6. PIPE COMPONENT OVERALL LEAK RATE SHALL NOT EXCEED 1 X 10^{-7} STD. CC/SEC AIR PER ANSI N14.5. THE O-RING USED FOR TESTING SHALL BE SHIPPED WITH THAT PARTICULAR PIPE COMPONENT.
7. TORQUE TO 40±5 LB-FT. LUBRICATED.
8. TORQUE TO 4-7 LB-FT. LUBRICATED.
9. ELECTRO-ETCH OR LASER MARK IDENTIFICATION USING 1/4 IN. HIGH CHARACTERS. MARKING REQUIRED TO BE PER CONTRACT REQUIREMENTS. LOCATE APPROXIMATELY WHERE SHOWN.
10. SURFACE FINISH APPLIES TO AN AREA WITH A 6.90 MAX. I.D. AND 7.80 MIN. O.D.
11. PIPE COMPONENT SHALL BE EQUIPPED WITH ONE FILTER VENT PER DOE/WIPP 11-3384.
12. OPTIONALLY, PARTS MAY BE MADE FROM 6 IN. X 150 LB. F304L ASTM A182 WELD NECK OR BLIND FLANGE.
13. PIPE FLANGE (ITEM 2) MAY BE ROUGH MACHINED AT PART LEVEL THEN FINAL MACHINED AT ASSEMBLY LEVEL (ITEM 1).
14. PIPE COMPONENT BODY (ITEM 3 PLATE/PIPE, ITEM 2 PIPE FLANGE, AND/OR ITEM 4 BOTTOM CAP) MAY BE FORMED FROM SINGLE PIECE IN LIEU OF WELDING. NWP ENGINEER APPROVAL REQUIRED.
15. ASSEMBLY LEVEL DIMENSIONAL REQUIREMENTS MUST BE MAINTAINED.
16. CARR-LANE OR NWP ENGINEER APPROVED EQUIVALENT.
17. PARKER SEALS OR NWP ENGINEER APPROVED EQUIVALENT.

REV	DESCRIPTION	DATE	APPROVED
2	REVISION		
1	REVISION	8/21/13	TS
0	REVISED FROM 165-F-028-W	9/18/12	TS

REVISIONS

UNLESS OTHERWISE SPECIFIED				
INTERPRET DWG PER ASME Y14.5-2009				
DIMENSIONS IN INCHES BASED ON 68 °F				
BREAK EDGES .03 MAX		FILLET RADII .03 MIN		
TOLERANCES				
FRACTION	1 PL DEC	2 PL DEC	3 PL DEC	ANGLES
±1/16	±.1	±.03	±.010	±.5°
THIRD ANGLE PROJECTION				FINISH 125

DRAFTER	DATE
CHECKER	DATE
ENGINEER	DATE
MANAGER	DATE
QUALITY ASSURANCE	DATE
APPROVED	DATE
APPROVED	DATE

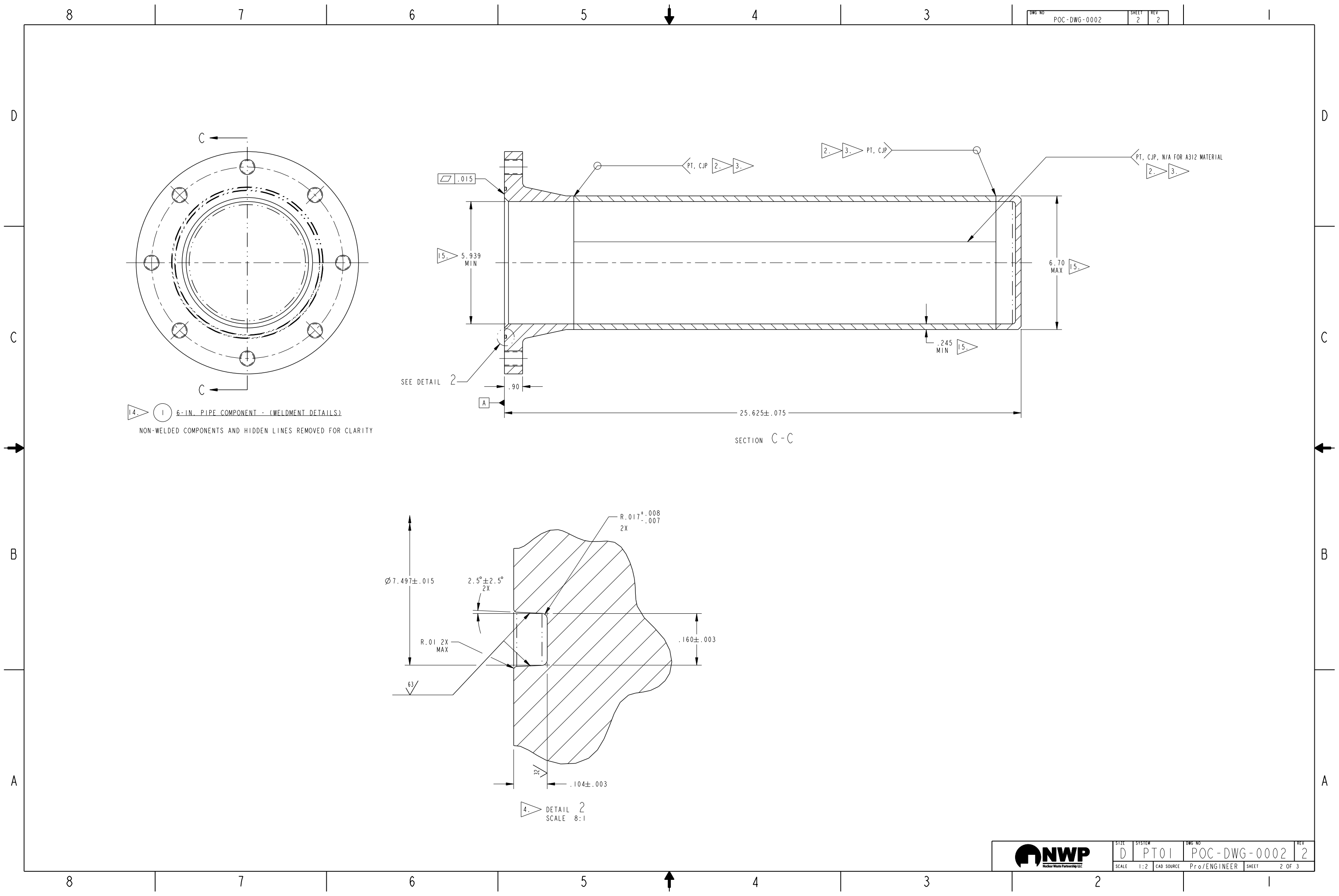
NWP Nuclear Waste Partnership LLC
Waste Isolation Pilot Plant — A URS-led partnership with B&W and AREVA — Carlsbad, NM

TITLE: 6-IN. PIPE COMPONENT

SIZE	SYSTEM	DWG NO	REV
D	PT01	POC-DWG-0002	2
SCALE 1:2	CAD SOURCE Pro/ENGINEER	SHEET 1 OF 3	

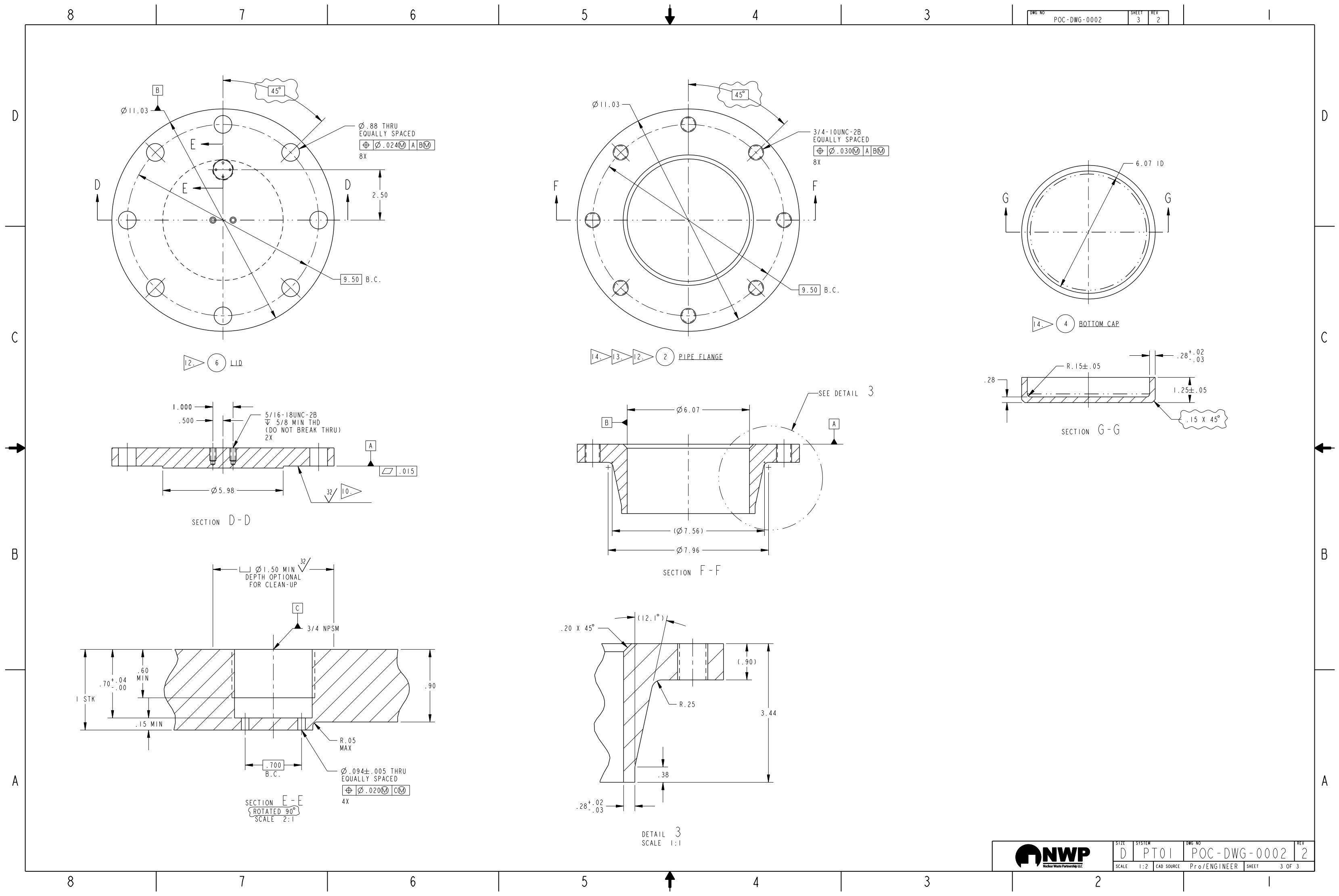
(a) Sheet 1

Figure A-2. 6-inch Pipe Component Drawing (McGonagill and Nevarez 2018a).



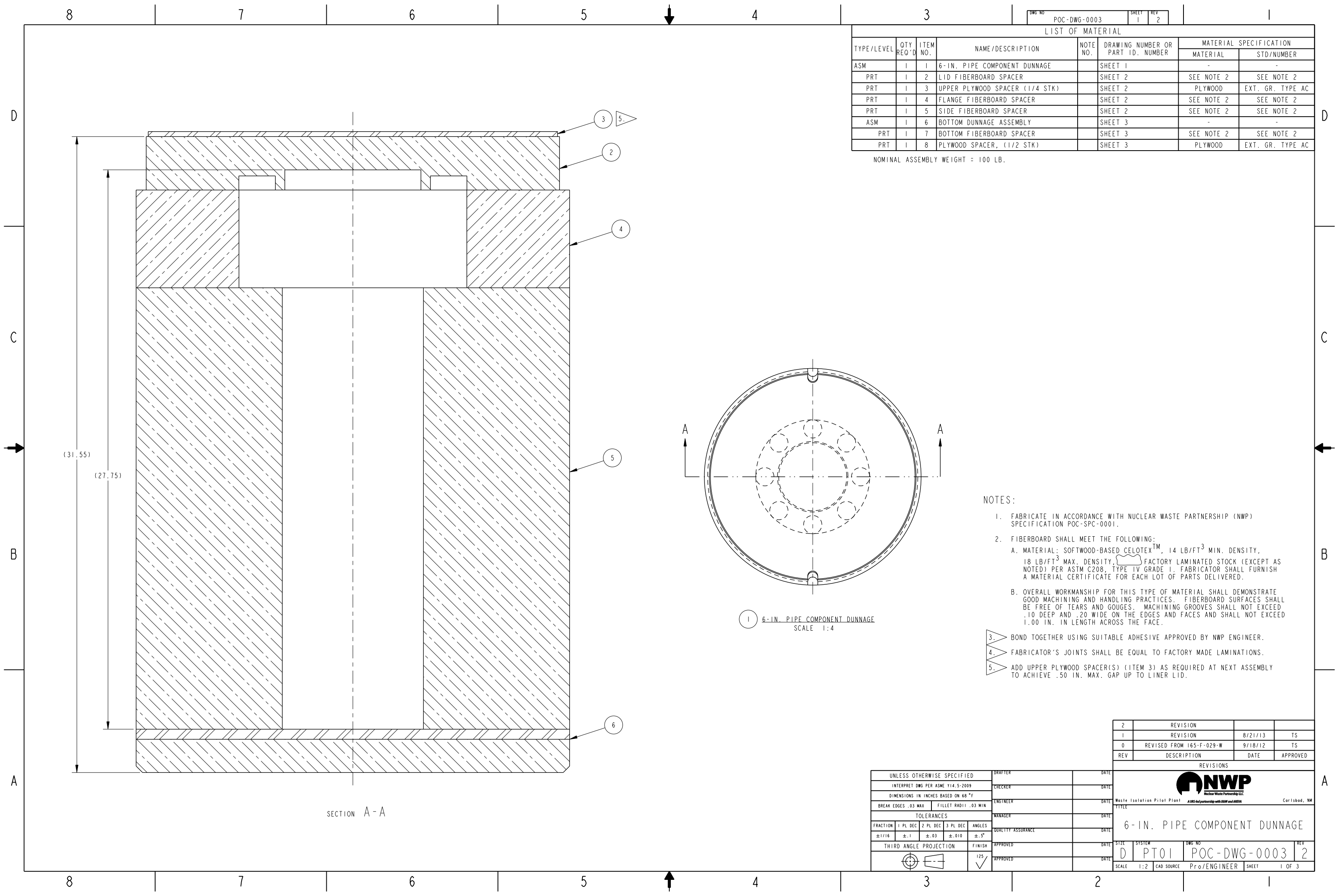
(b) Sheet 2

Figure A-2. 60-inch Pipe Component Drawing (McGonagill and Nevarez 2018a) (continued).



(c) Sheet 3

Figure A-2. 6-inch Pipe Component Drawing (McGonagill and Nevarez 2018a) (continued).



DWG NO	SHEET	REV
POC-DWG-0003	1	2

LIST OF MATERIAL

TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DRAWING NUMBER OR PART ID. NUMBER	MATERIAL SPECIFICATION	
						MATERIAL	STD/NUMBER
ASM	1	1	6-IN. PIPE COMPONENT DUNNAGE		SHEET 1	-	-
PRT	1	2	LID FIBERBOARD SPACER		SHEET 2	SEE NOTE 2	SEE NOTE 2
PRT	1	3	UPPER PLYWOOD SPACER (1/4 STK)		SHEET 2	PLYWOOD	EXT. GR. TYPE AC
PRT	1	4	FLANGE FIBERBOARD SPACER		SHEET 2	SEE NOTE 2	SEE NOTE 2
PRT	1	5	SIDE FIBERBOARD SPACER		SHEET 2	SEE NOTE 2	SEE NOTE 2
ASM	1	6	BOTTOM DUNNAGE ASSEMBLY		SHEET 3	-	-
PRT	1	7	BOTTOM FIBERBOARD SPACER		SHEET 3	SEE NOTE 2	SEE NOTE 2
PRT	1	8	PLYWOOD SPACER, (1/2 STK)		SHEET 3	PLYWOOD	EXT. GR. TYPE AC

NOMINAL ASSEMBLY WEIGHT = 100 LB.

NOTES:

1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001.

2. FIBERBOARD SHALL MEET THE FOLLOWING:

   A. MATERIAL: SOFTWOOD-BASED CELOTEX™, 14 LB/FT^3 MIN. DENSITY, 18 LB/FT^3 MAX. DENSITY, FACTORY LAMINATED STOCK (EXCEPT AS NOTED) PER ASTM C208, TYPE IV GRADE 1. FABRICATOR SHALL FURNISH A MATERIAL CERTIFICATE FOR EACH LOT OF PARTS DELIVERED.

   B. OVERALL WORKMANSHIP FOR THIS TYPE OF MATERIAL SHALL DEMONSTRATE GOOD MACHINING AND HANDLING PRACTICES. FIBERBOARD SURFACES SHALL BE FREE OF TEARS AND GOUGES. MACHINING GROOVES SHALL NOT EXCEED .10 DEEP AND .20 WIDE ON THE EDGES AND FACES AND SHALL NOT EXCEED 1.00 IN. IN LENGTH ACROSS THE FACE.

3. BOND TOGETHER USING SUITABLE ADHESIVE APPROVED BY NWP ENGINEER.

4. FABRICATOR'S JOINTS SHALL BE EQUAL TO FACTORY MADE LAMINATIONS.

5. ADD UPPER PLYWOOD SPACER(S) (ITEM 3) AS REQUIRED AT NEXT ASSEMBLY TO ACHIEVE .50 IN. MAX. GAP UP TO LINER LID.

REV	DESCRIPTION	DATE	APPROVED
2	REVISION		
1	REVISION	8/21/13	TS
0	REVISED FROM 165-F-029-W	9/18/12	TS

REVISIONS

UNLESS OTHERWISE SPECIFIED				
INTERPRET DWG PER ASME Y14.5-2009				
DIMENSIONS IN INCHES BASED ON 68 °F				
BREAK EDGES .03 MAX	FILLET RADII .03 MIN			
TOLERANCES				
FRACTION	1 PL DEC	2 PL DEC	3 PL DEC	ANGLES
±1/16	±.1	±.03	±.010	±.5°
THIRD ANGLE PROJECTION				FINISH 125

DRAFTER	DATE
CHECKER	DATE
ENGINEER	DATE
MANAGER	DATE
QUALITY ASSURANCE	DATE
APPROVED	DATE
APPROVED	DATE

NWP Nuclear Waste Partnership LLC
Waste Isolation Pilot Plant A URS-led partnership with B&W and AREVA Carlsbad, NM

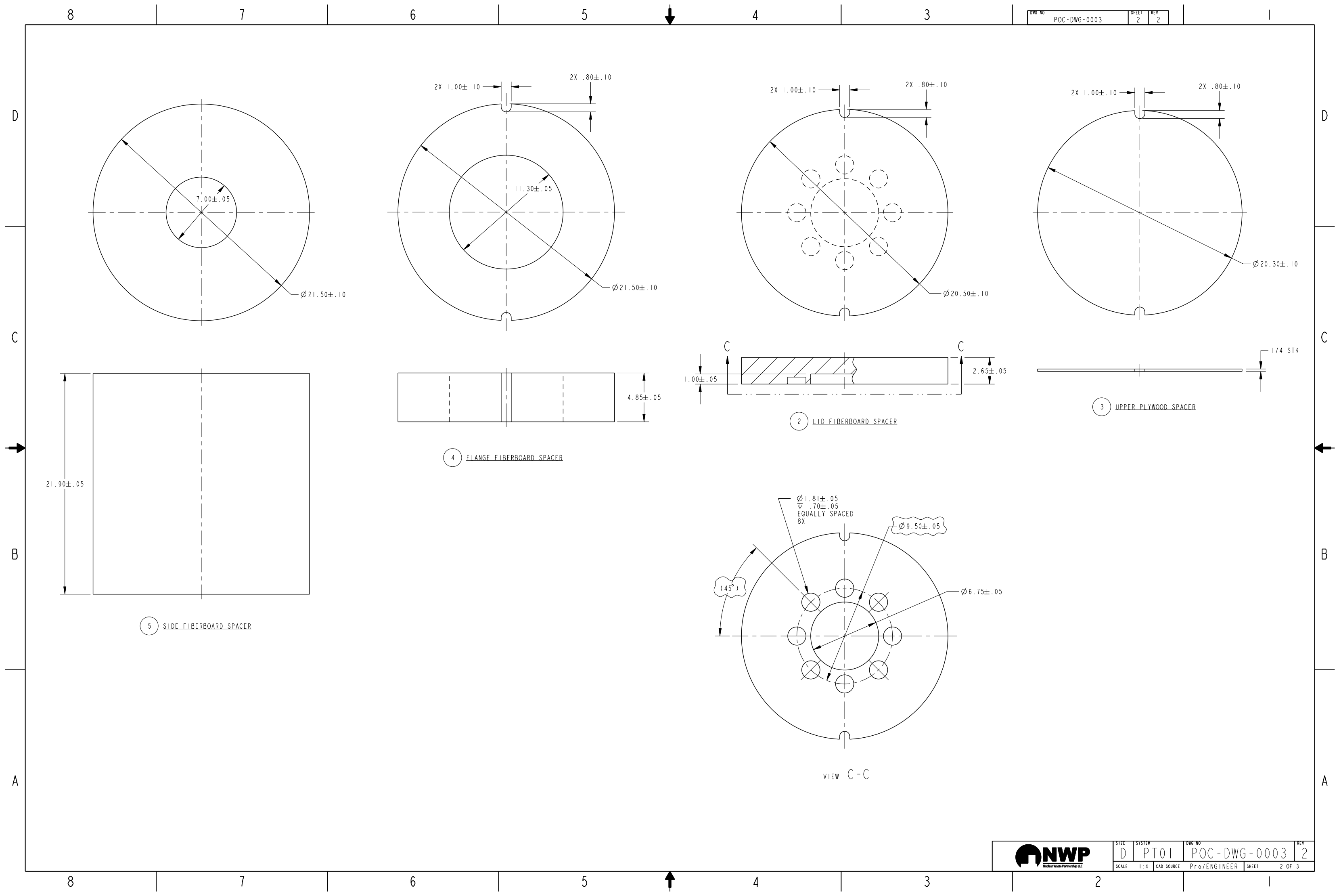
TITLE 6-IN. PIPE COMPONENT DUNNAGE

SIZE	SYSTEM	DWG NO	REV
D	PT01	POC-DWG-0003	2

SCALE 1:2 CAD SOURCE Pro/ENGINEER SHEET 1 OF 3

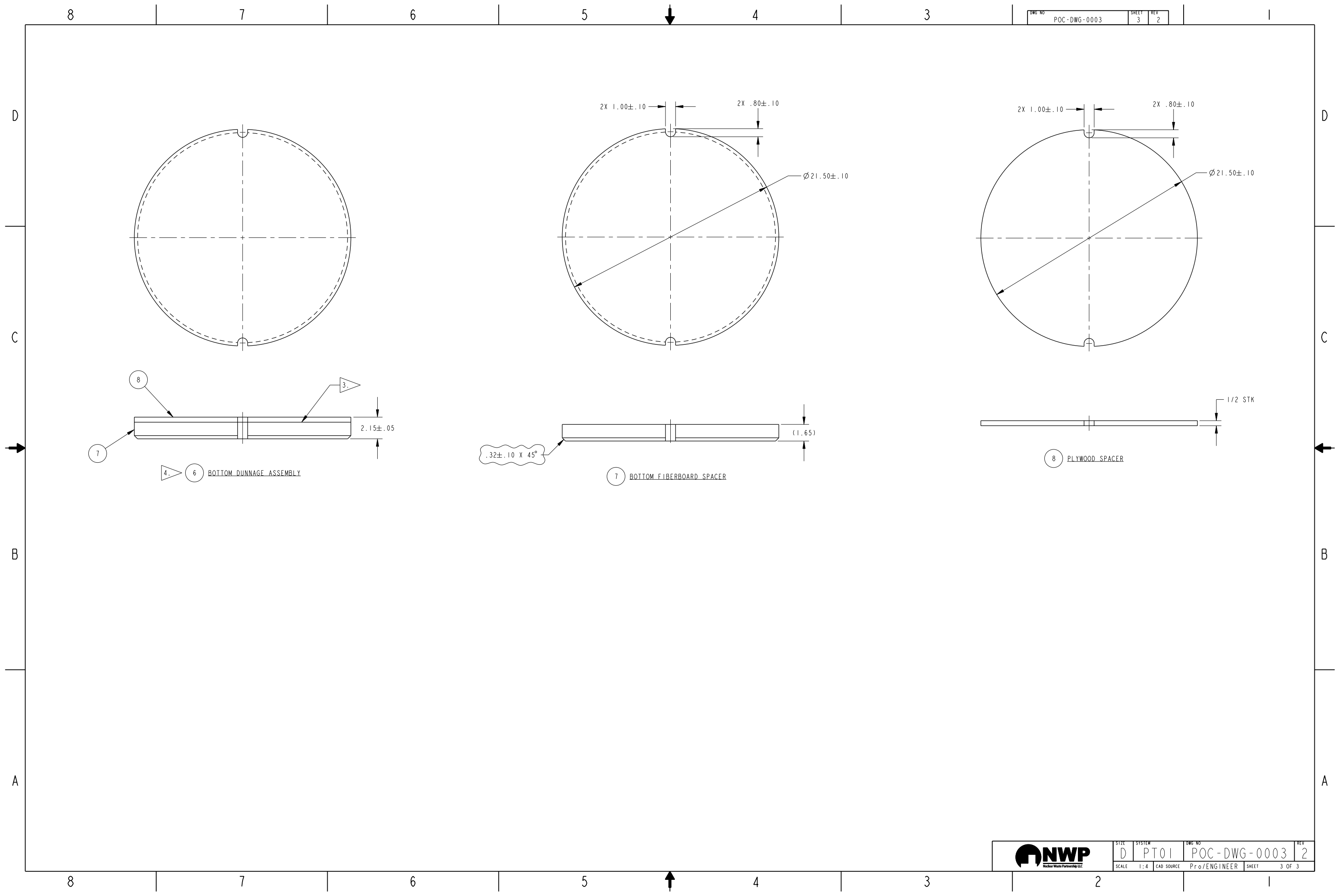
(a) Sheet 1

Figure A-3. 6-inch Pipe Component Dunnage Drawing (McGonagill and Nevarez 2018b).



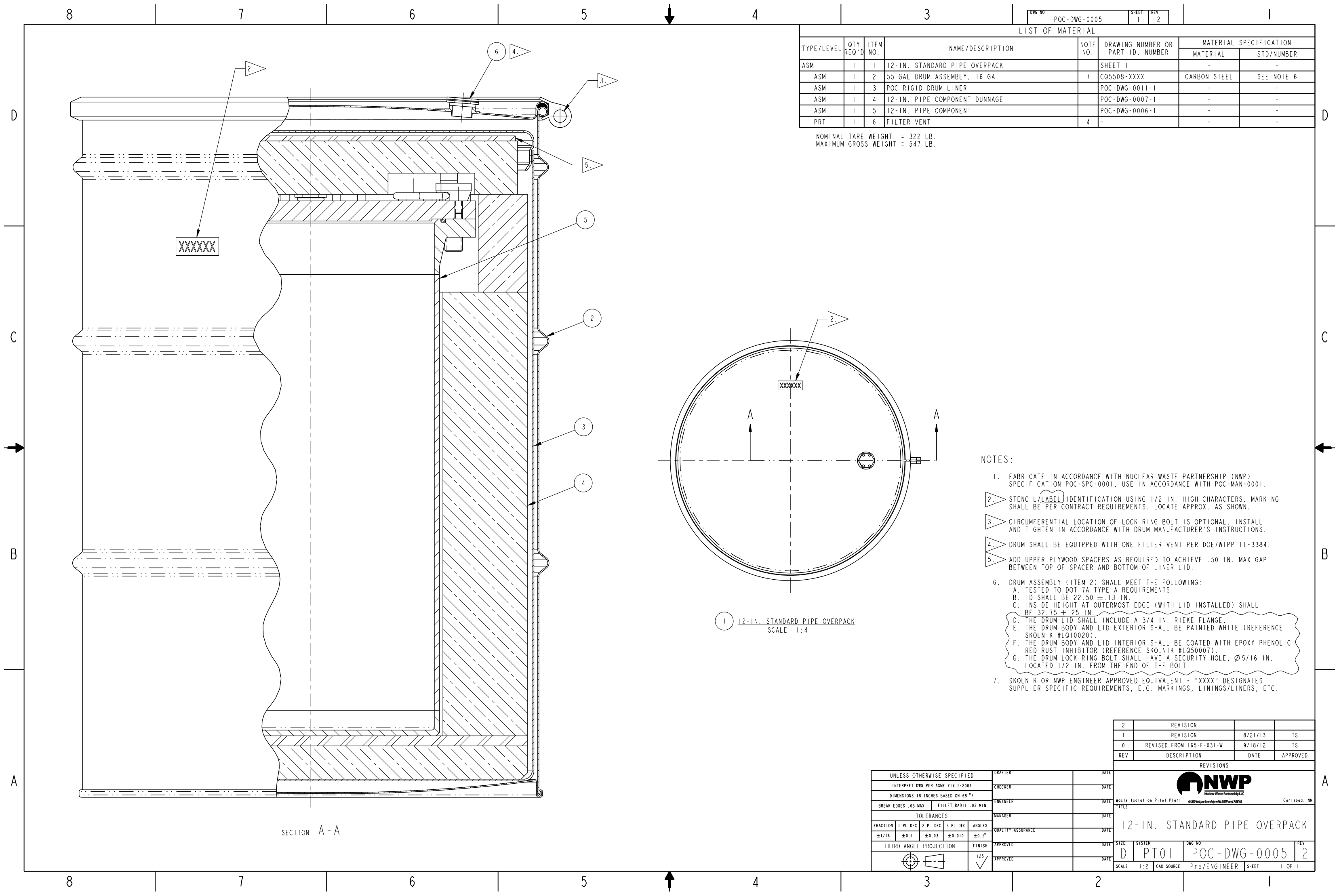
(b) **Sheet 2**

Figure A-3. 6-inch Pipe Component Dunnage Drawing (McGonagill and Nevarez 2018b) (continued).



(c) Sheet 3

Figure A-3. 6-inch Pipe Component Dunnage Drawing (McGonagill and Nevarez 2018b) (continued).


DWG NO POC-DWG-0005 SHEET 1 REV 2
SECTION A-A
1 12-IN. STANDARD PIPE OVERPACK SCALE 1:4
XXXXXX
A
A
1
2
3
4
5
6
2.
3.
4.
5.

LIST OF MATERIAL

TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DRAWING NUMBER OR PART ID. NUMBER	MATERIAL SPECIFICATION	
						MATERIAL	STD/NUMBER
ASM	1	1	12-IN. STANDARD PIPE OVERPACK		SHEET 1	-	-
ASM	1	2	55 GAL DRUM ASSEMBLY, 16 GA.	7	CQ5508-XXXX	CARBON STEEL	SEE NOTE 6
ASM	1	3	POC RIGID DRUM LINER		POC-DWG-0011-1	-	-
ASM	1	4	12-IN. PIPE COMPONENT DUNNAGE		POC-DWG-0007-1	-	-
ASM	1	5	12-IN. PIPE COMPONENT		POC-DWG-0006-1	-	-
PRT	1	6	FILTER VENT	4	-	-	-

NOMINAL TARE WEIGHT = 322 LB.
MAXIMUM GROSS WEIGHT = 547 LB.

NOTES:

1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001. USE IN ACCORDANCE WITH POC-MAN-0001.

2. STENCIL/LABEL IDENTIFICATION USING 1/2 IN. HIGH CHARACTERS. MARKING SHALL BE PER CONTRACT REQUIREMENTS. LOCATE APPROX. AS SHOWN.

3. CIRCUMFERENTIAL LOCATION OF LOCK RING BOLT IS OPTIONAL. INSTALL AND TIGHTEN IN ACCORDANCE WITH DRUM MANUFACTURER'S INSTRUCTIONS.

4. DRUM SHALL BE EQUIPPED WITH ONE FILTER VENT PER DOE/WIPP 11-3384.

5. ADD UPPER PLYWOOD SPACERS AS REQUIRED TO ACHIEVE .50 IN. MAX GAP BETWEEN TOP OF SPACER AND BOTTOM OF LINER LID.

6. DRUM ASSEMBLY (ITEM 2) SHALL MEET THE FOLLOWING:
   A. TESTED TO DOT 7A TYPE A REQUIREMENTS.
   B. ID SHALL BE 22.50 $\pm$.13 IN.
   C. INSIDE HEIGHT AT OUTERMOST EDGE (WITH LID INSTALLED) SHALL BE 32.75 $\pm$.25 IN.
   D. THE DRUM LID SHALL INCLUDE A 3/4 IN. RIEKE FLANGE.
   E. THE DRUM BODY AND LID EXTERIOR SHALL BE PAINTED WHITE (REFERENCE SKOLNIK #LQ10020).
   F. THE DRUM BODY AND LID INTERIOR SHALL BE COATED WITH EPOXY PHENOLIC RED RUST INHIBITOR (REFERENCE SKOLNIK #LQ50007).
   G. THE DRUM LOCK RING BOLT SHALL HAVE A SECURITY HOLE, Ø5/16 IN. LOCATED 1/2 IN. FROM THE END OF THE BOLT.

7. SKOLNIK OR NWP ENGINEER APPROVED EQUIVALENT - "XXXX" DESIGNATES SUPPLIER SPECIFIC REQUIREMENTS, E.G. MARKINGS, LININGS/LINERS, ETC.

REV	DESCRIPTION	DATE	APPROVED
2	REVISION		
1	REVISION	8/21/13	TS
0	REVISED FROM 165-F-031-W	9/18/12	TS

REVISIONS

UNLESS OTHERWISE SPECIFIED				
INTERPRET DWG PER ASME Y14.5-2009				
DIMENSIONS IN INCHES BASED ON 68 °F				
BREAK EDGES .03 MAX	FILLET RADII .03 MIN			
TOLERANCES				
FRACTION	1 PL DEC	2 PL DEC	3 PL DEC	ANGLES
±1/16	±0.1	±0.03	±0.010	±0.5°
THIRD ANGLE PROJECTION	FINISH 125			

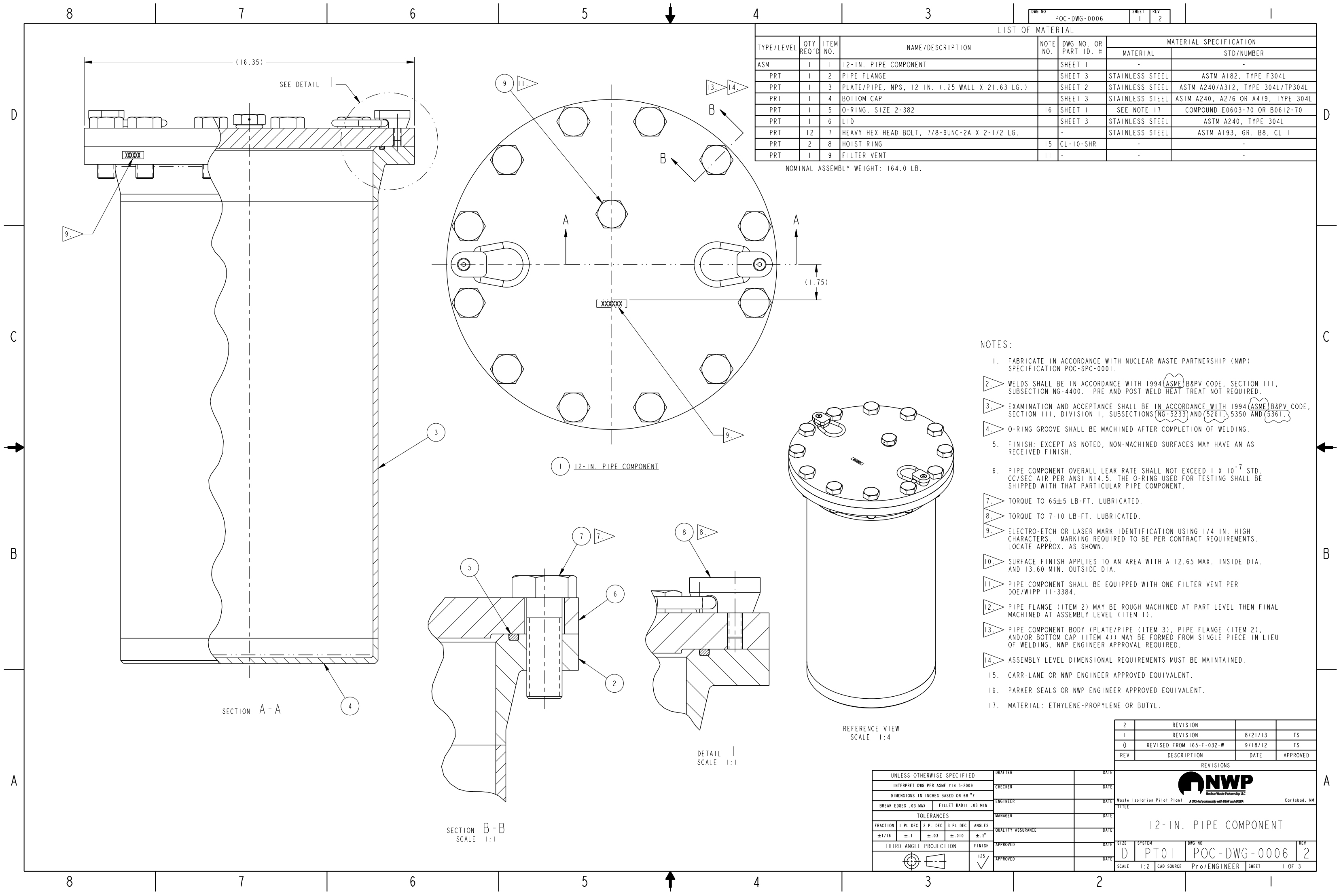
DRAFTER	DATE
CHECKER	DATE
ENGINEER	DATE
MANAGER	DATE
QUALITY ASSURANCE	DATE
APPROVED	DATE
APPROVED	DATE

NWP Nuclear Waste Partnership LLC
Waste Isolation Pilot Plant — A URS-led partnership with B&W and AREVA — Carlsbad, NM

TITLE: 12-IN. STANDARD PIPE OVERPACK

SIZE	SYSTEM	DWG NO	REV
D	PT01	POC-DWG-0005	2
SCALE 1:2	CAD SOURCE Pro/ENGINEER	SHEET 1 OF 1	

Figure A-4. 12-inch Standard Pipe Overpack Assembly Drawing (McGonagill and Nevarez 2018c).



DWG NO	SHEET	REV
POC-DWG-0006	1	2

LIST OF MATERIAL

TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DWG NO. OR PART ID. #	MATERIAL SPECIFICATION	
						MATERIAL	STD/NUMBER
ASM	1	1	12-IN. PIPE COMPONENT		SHEET 1	-	-
PRT	1	2	PIPE FLANGE		SHEET 3	STAINLESS STEEL	ASTM A182, TYPE F304L
PRT	1	3	PLATE/PIPE, NPS, 12 IN. (.25 WALL X 21.63 LG.)		SHEET 2	STAINLESS STEEL	ASTM A240/A312, TYPE 304L/TP304L
PRT	1	4	BOTTOM CAP		SHEET 3	STAINLESS STEEL	ASTM A240, A276 OR A479, TYPE 304L
PRT	1	5	O-RING, SIZE 2-382	16	SHEET 1	SEE NOTE 17	COMPOUND E0603-70 OR B0612-70
PRT	1	6	LID		SHEET 3	STAINLESS STEEL	ASTM A240, TYPE 304L
PRT	12	7	HEAVY HEX HEAD BOLT, 7/8-9UNC-2A X 2-1/2 LG.		-	STAINLESS STEEL	ASTM A193, GR. B8, CL 1
PRT	2	8	HOIST RING	15	CL-10-SHR	-	-
PRT	1	9	FILTER VENT	11	-	-	-

NOMINAL ASSEMBLY WEIGHT: 164.0 LB.

NOTES:

1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001.
2. WELDS SHALL BE IN ACCORDANCE WITH 1994 ASME B&PV CODE, SECTION III, SUBSECTION NG-4400. PRE AND POST WELD HEAT TREAT NOT REQUIRED.
3. EXAMINATION AND ACCEPTANCE SHALL BE IN ACCORDANCE WITH 1994 ASME B&PV CODE, SECTION III, DIVISION 1, SUBSECTIONS NG-5233 AND 5261, 5350 AND 5361.
4. O-RING GROOVE SHALL BE MACHINED AFTER COMPLETION OF WELDING.
5. FINISH: EXCEPT AS NOTED, NON-MACHINED SURFACES MAY HAVE AN AS RECEIVED FINISH.
6. PIPE COMPONENT OVERALL LEAK RATE SHALL NOT EXCEED 1 X 10^{-7} STD. CC/SEC AIR PER ANSI N14.5. THE O-RING USED FOR TESTING SHALL BE SHIPPED WITH THAT PARTICULAR PIPE COMPONENT.
7. TORQUE TO 65±5 LB-FT. LUBRICATED.
8. TORQUE TO 7-10 LB-FT. LUBRICATED.
9. ELECTRO-ETCH OR LASER MARK IDENTIFICATION USING 1/4 IN. HIGH CHARACTERS. MARKING REQUIRED TO BE PER CONTRACT REQUIREMENTS. LOCATE APPROX. AS SHOWN.
10. SURFACE FINISH APPLIES TO AN AREA WITH A 12.65 MAX. INSIDE DIA. AND 13.60 MIN. OUTSIDE DIA.
11. PIPE COMPONENT SHALL BE EQUIPPED WITH ONE FILTER VENT PER DOE/WIPP 11-3384.
12. PIPE FLANGE (ITEM 2) MAY BE ROUGH MACHINED AT PART LEVEL THEN FINAL MACHINED AT ASSEMBLY LEVEL (ITEM 1).
13. PIPE COMPONENT BODY (PLATE/PIPE (ITEM 3), PIPE FLANGE (ITEM 2), AND/OR BOTTOM CAP (ITEM 4)) MAY BE FORMED FROM SINGLE PIECE IN LIEU OF WELDING. NWP ENGINEER APPROVAL REQUIRED.
14. ASSEMBLY LEVEL DIMENSIONAL REQUIREMENTS MUST BE MAINTAINED.
15. CARR-LANE OR NWP ENGINEER APPROVED EQUIVALENT.
16. PARKER SEALS OR NWP ENGINEER APPROVED EQUIVALENT.
17. MATERIAL: ETHYLENE-PROPYLENE OR BUTYL.

REVISIONS

REV	DESCRIPTION	DATE	APPROVED
2	REVISION		
1	REVISION	8/21/13	TS
0	REVISED FROM 165-F-032-W	9/18/12	TS

UNLESS OTHERWISE SPECIFIED				
INTERPRET DWG PER ASME Y14.5-2009				
DIMENSIONS IN INCHES BASED ON 68 °F				
BREAK EDGES .03 MAX		FILLET RADII .03 MIN		
TOLERANCES				
FRACTION	1 PL DEC	2 PL DEC	3 PL DEC	ANGLES
±1/16	±.1	±.03	±.010	±.5°
THIRD ANGLE PROJECTION				FINISH 125

DRAFTER	DATE
CHECKER	DATE
ENGINEER	DATE
MANAGER	DATE
QUALITY ASSURANCE	DATE
APPROVED	DATE
APPROVED	DATE

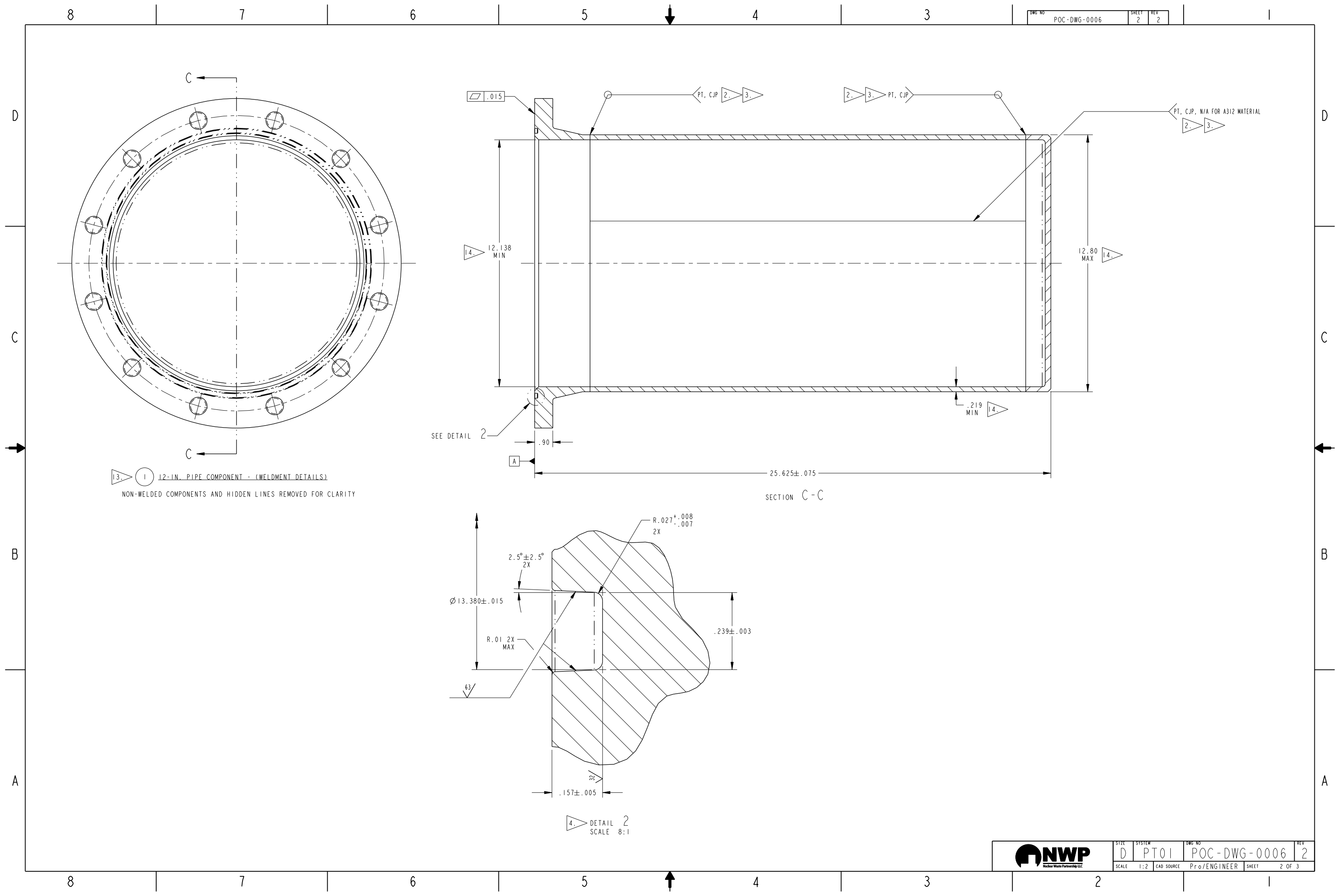
NWP Nuclear Waste Partnership LLC
Waste Isolation Pilot Plant — A URS-led partnership with B&W and AREVA — Carlsbad, NM

TITLE: 12-IN. PIPE COMPONENT

SIZE	SYSTEM	DWG NO	REV
D	PT01	POC-DWG-0006	2
SCALE 1:2	CAD SOURCE Pro/ENGINEER	SHEET 1 OF 3	

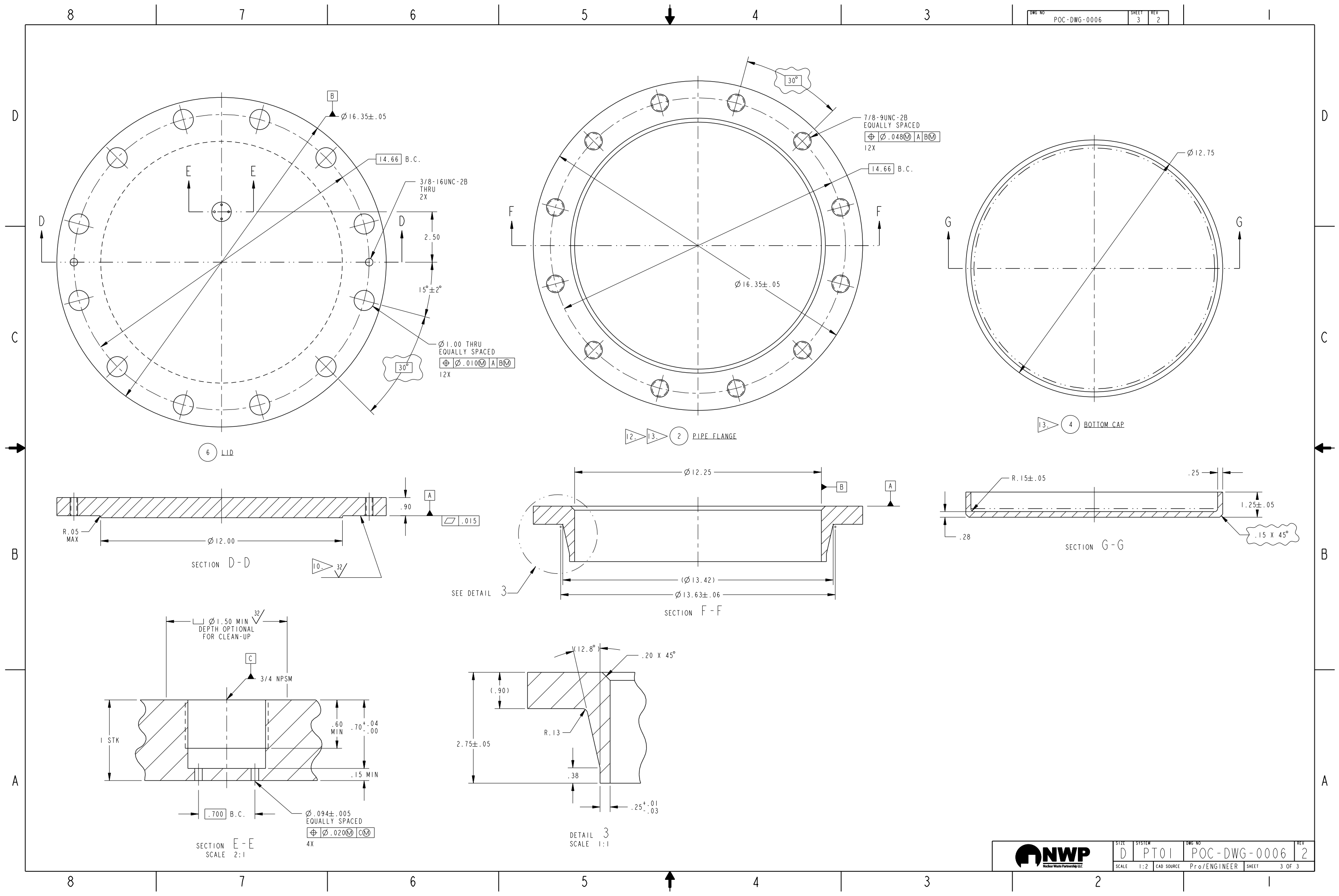
(a) Sheet 1

Figure A-5. 12-inch Pipe Component Drawing (McGonagill and Nevarez 2018a).



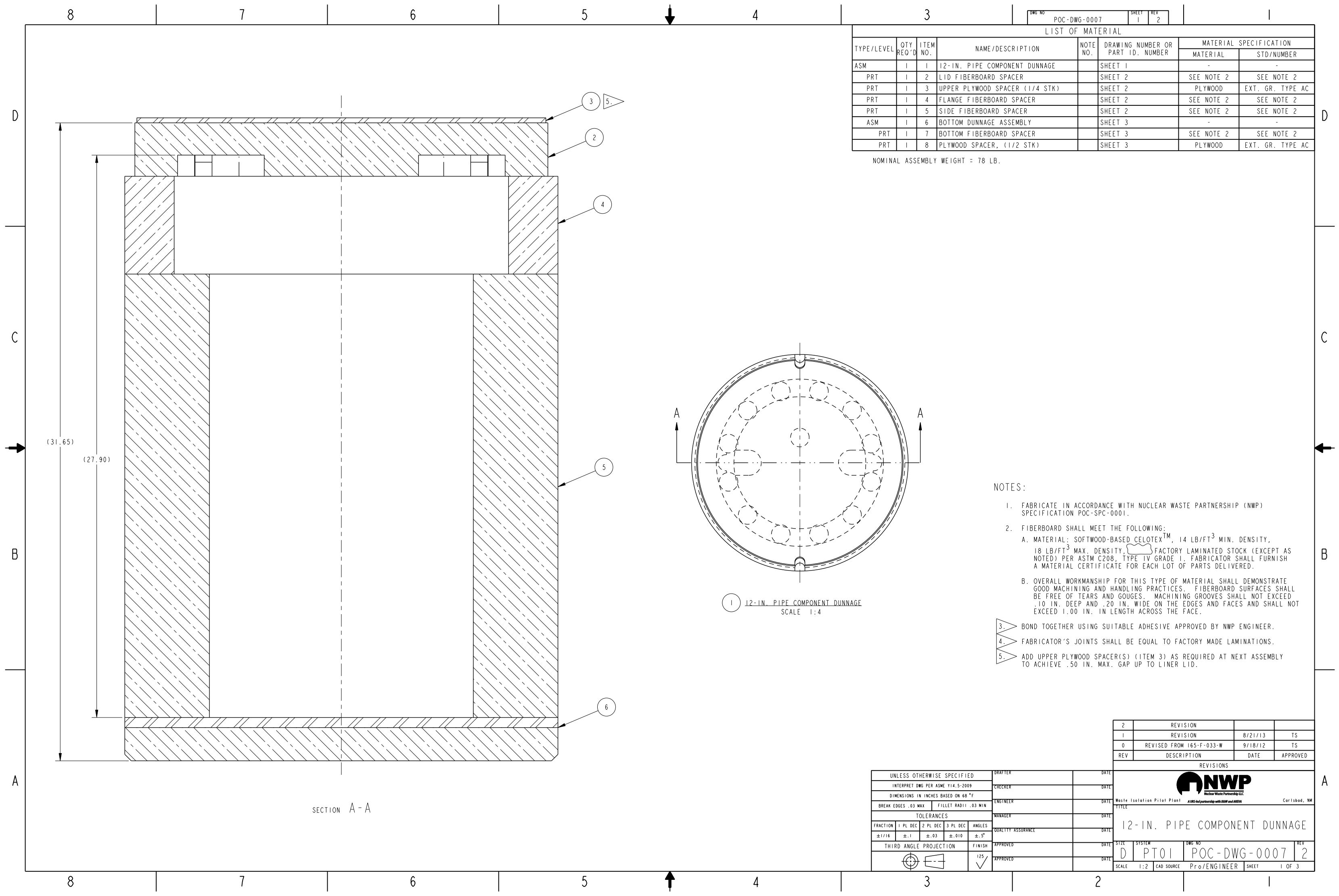
(b) **Sheet 2**

Figure A-5. 12-inch Pipe Component Drawing (McGonagill and Nevarez 2018a) (continued).



(c) Sheet 3

Figure A-5. 12" Pipe Component Drawing (McGonagill and Nevarez 2018a) (continued).


DWG NO POC-DWG-0007 SHEET 1 REV 2
(31.65)
(27.90)
SECTION A-A
A
A
1 12-IN. PIPE COMPONENT DUNNAGE
SCALE 1:4

LIST OF MATERIAL

TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DRAWING NUMBER OR PART ID. NUMBER	MATERIAL SPECIFICATION	
						MATERIAL	STD/NUMBER
ASM	1	1	12-IN. PIPE COMPONENT DUNNAGE		SHEET 1	-	-
PRT	1	2	LID FIBERBOARD SPACER		SHEET 2	SEE NOTE 2	SEE NOTE 2
PRT	1	3	UPPER PLYWOOD SPACER (1/4 STK)		SHEET 2	PLYWOOD	EXT. GR. TYPE AC
PRT	1	4	FLANGE FIBERBOARD SPACER		SHEET 2	SEE NOTE 2	SEE NOTE 2
PRT	1	5	SIDE FIBERBOARD SPACER		SHEET 2	SEE NOTE 2	SEE NOTE 2
ASM	1	6	BOTTOM DUNNAGE ASSEMBLY		SHEET 3	-	-
PRT	1	7	BOTTOM FIBERBOARD SPACER		SHEET 3	SEE NOTE 2	SEE NOTE 2
PRT	1	8	PLYWOOD SPACER, (1/2 STK)		SHEET 3	PLYWOOD	EXT. GR. TYPE AC

NOMINAL ASSEMBLY WEIGHT = 78 LB.

NOTES:

1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001.

2. FIBERBOARD SHALL MEET THE FOLLOWING:
   A. MATERIAL: SOFTWOOD-BASED CELOTEX^{TM}, 14 LB/FT^3 MIN. DENSITY, 18 LB/FT^3 MAX. DENSITY, FACTORY LAMINATED STOCK (EXCEPT AS NOTED) PER ASTM C208, TYPE IV GRADE 1. FABRICATOR SHALL FURNISH A MATERIAL CERTIFICATE FOR EACH LOT OF PARTS DELIVERED.
   B. OVERALL WORKMANSHIP FOR THIS TYPE OF MATERIAL SHALL DEMONSTRATE GOOD MACHINING AND HANDLING PRACTICES. FIBERBOARD SURFACES SHALL BE FREE OF TEARS AND GOUGES. MACHINING GROOVES SHALL NOT EXCEED .10 IN. DEEP AND .20 IN. WIDE ON THE EDGES AND FACES AND SHALL NOT EXCEED 1.00 IN. IN LENGTH ACROSS THE FACE.

3. BOND TOGETHER USING SUITABLE ADHESIVE APPROVED BY NWP ENGINEER.

4. FABRICATOR'S JOINTS SHALL BE EQUAL TO FACTORY MADE LAMINATIONS.

5. ADD UPPER PLYWOOD SPACER(S) (ITEM 3) AS REQUIRED AT NEXT ASSEMBLY TO ACHIEVE .50 IN. MAX. GAP UP TO LINER LID.

REV	DESCRIPTION	DATE	APPROVED
2	REVISION		
1	REVISION	8/21/13	TS
0	REVISED FROM 165-F-033-W	9/18/12	TS

REVISIONS

UNLESS OTHERWISE SPECIFIED					DRAFTER	DATE
INTERPRET DWG PER ASME Y14.5-2009					CHECKER	DATE
DIMENSIONS IN INCHES BASED ON 68 °F					ENGINEER	DATE
BREAK EDGES .03 MAX		FILLET RADII .03 MIN			MANAGER	DATE
TOLERANCES					QUALITY ASSURANCE	DATE
FRACTION	1 PL DEC	2 PL DEC	3 PL DEC	ANGLES	APPROVED	DATE
±1/16	±.1	±.03	±.010	±.5°	APPROVED	DATE
THIRD ANGLE PROJECTION				FINISH 125		

NWP Nuclear Waste Partnership LLC
Waste Isolation Pilot Plant — A URS-led partnership with B&W and AREVA — Carlsbad, NM

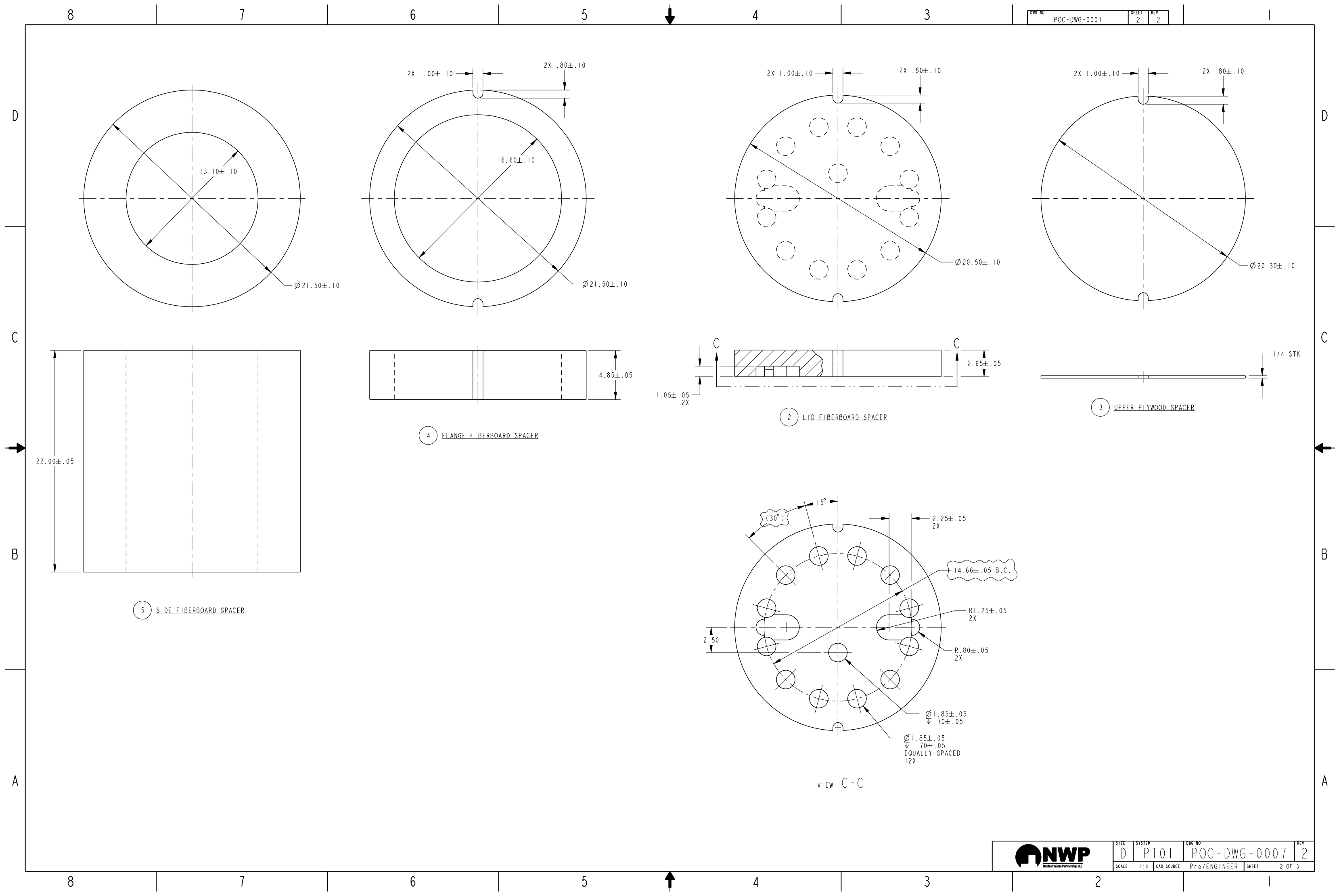
TITLE: 12-IN. PIPE COMPONENT DUNNAGE

SIZE	SYSTEM	DWG NO	REV
D	PT01	POC-DWG-0007	2

SCALE 1:2 | CAD SOURCE Pro/ENGINEER | SHEET 1 OF 3

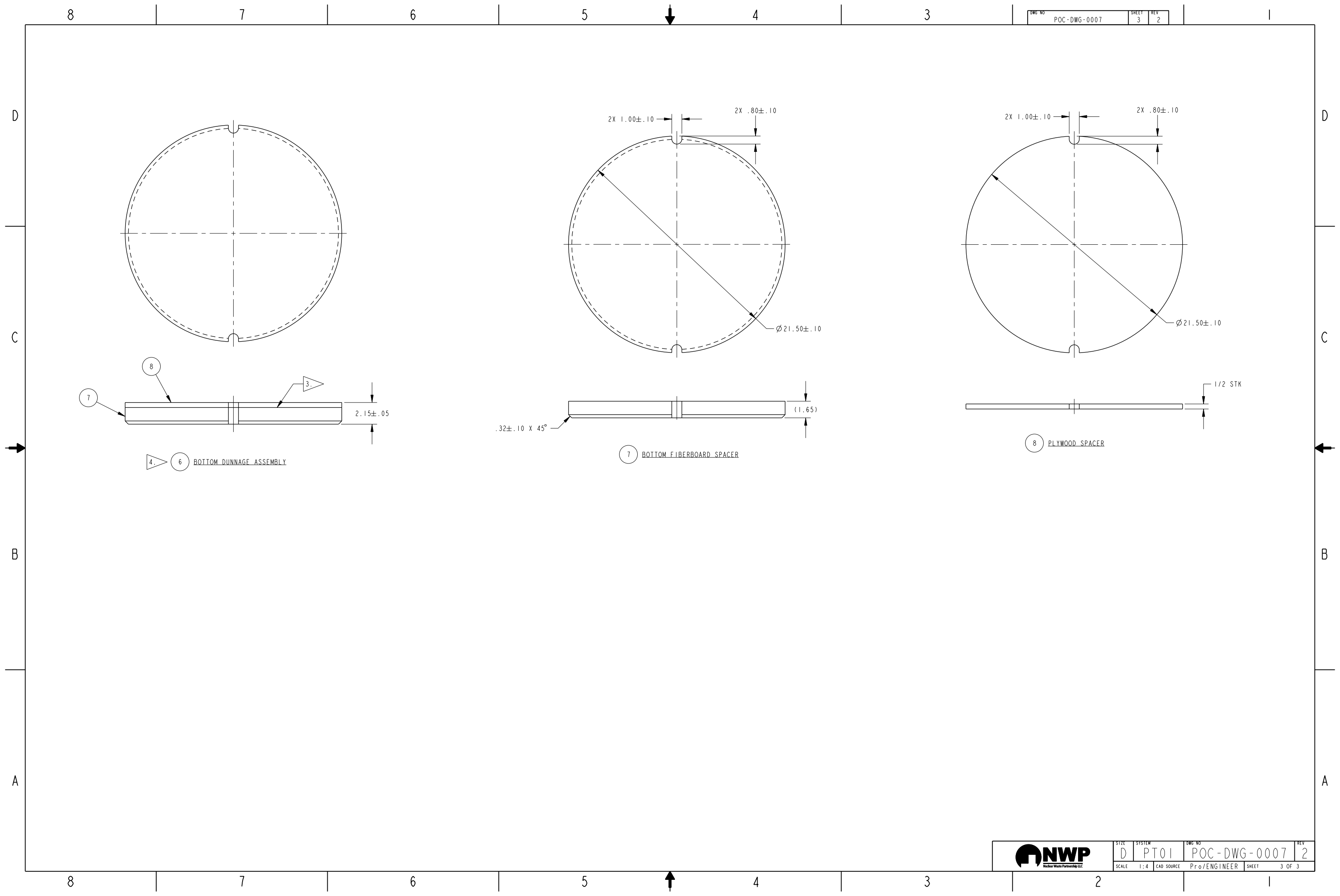
(a) Sheet 1

Figure A-6. 12-inch Pipe Component Dunnage Drawing (McGonagill and Nevarez 2018b).



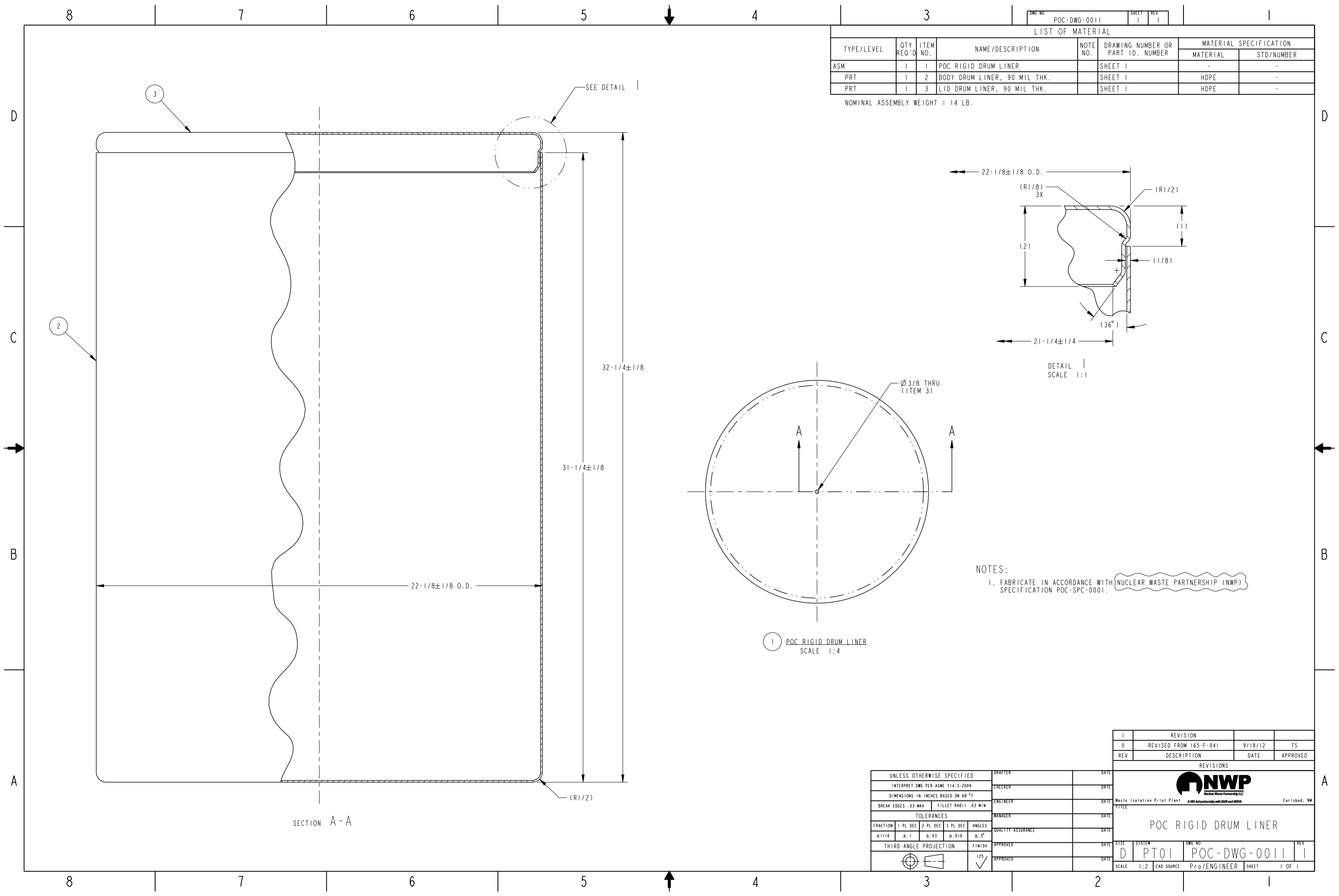
(b) Sheet 2

Figure A-6. 12-inch Pipe Component Dunnage Drawing (McGonagill and Nevarez 2018b) (continued).



(c) Sheet 3

Figure A-6. 12-inch Pipe Component Dunnage Drawing (McGonagill and Nevarez 2018b) (continued).

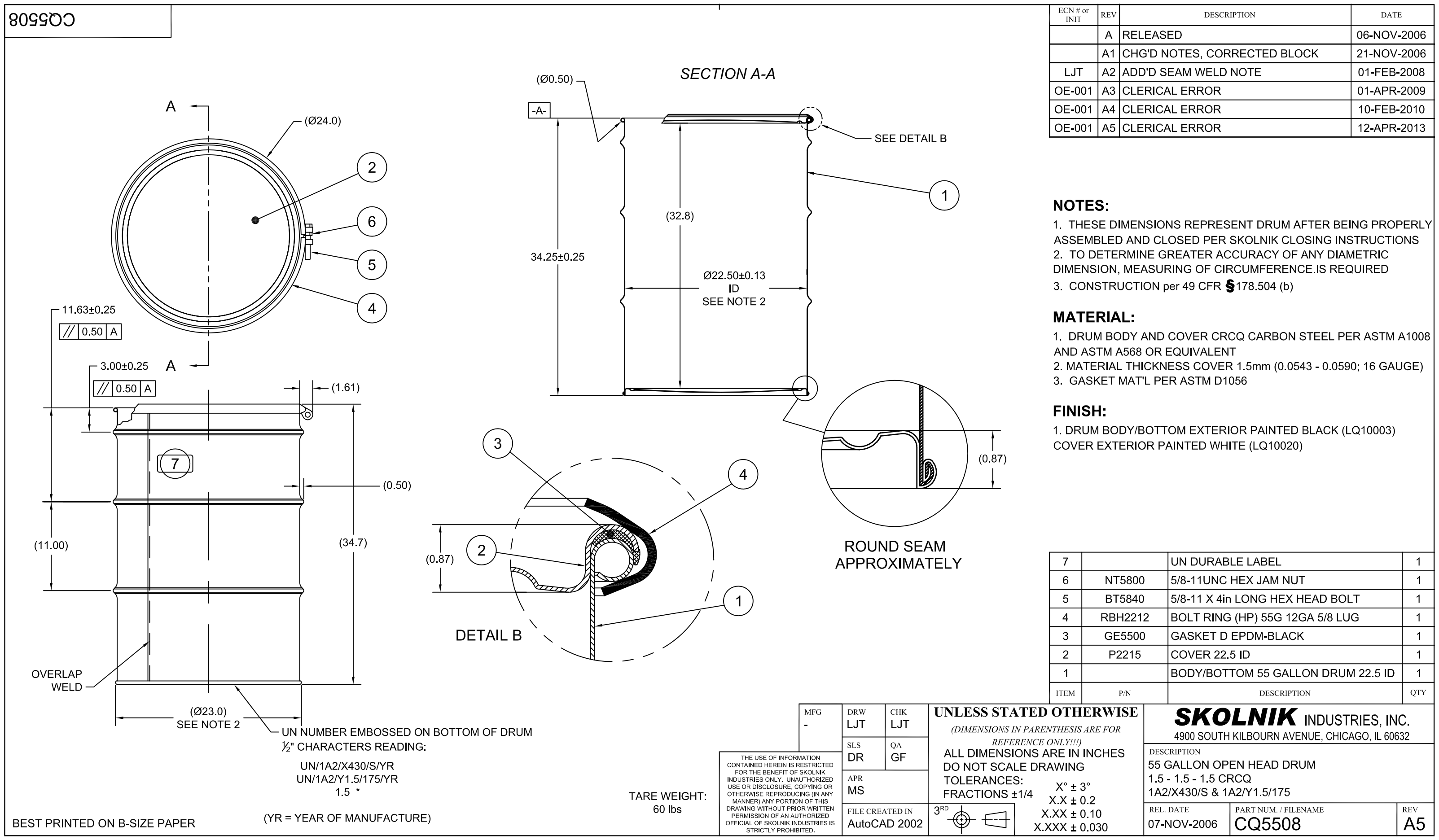


TYPE/LEVEL	QTY REQ'D	ITEM NO.	NAME/DESCRIPTION	NOTE NO.	DRAWING NUMBER OR PART ID. NUMBER	MATERIAL SPECIFICATION MATERIAL	MATERIAL SPECIFICATION STD/NUMBER
ASM	1	1	POC RIGID DRUM LINER		SHEET 1	-	-
PRT	1	2	BODY DRUM LINER, 90 MIL THK.		SHEET 1	HDPE	-
PRT	1	3	LID DRUM LINER, 90 MIL THK.		SHEET 1	HDPE	-

NOMINAL ASSEMBLY WEIGHT = 14 LB.

NOTES:
1. FABRICATE IN ACCORDANCE WITH NUCLEAR WASTE PARTNERSHIP (NWP) SPECIFICATION POC-SPC-0001.

Figure A-6. POC "Rigid" Drum Liner (McGonagill and Nevarez 2018d).



ECN # or INIT	REV	DESCRIPTION	DATE
	A	RELEASED	06-NOV-2006
	A1	CHG'D NOTES, CORRECTED BLOCK	21-NOV-2006
LJT	A2	ADD'D SEAM WELD NOTE	01-FEB-2008
OE-001	A3	CLERICAL ERROR	01-APR-2009
OE-001	A4	CLERICAL ERROR	10-FEB-2010
OE-001	A5	CLERICAL ERROR	12-APR-2013

NOTES:

1. THESE DIMENSIONS REPRESENT DRUM AFTER BEING PROPERLY ASSEMBLED AND CLOSED PER SKOLNIK CLOSING INSTRUCTIONS
2. TO DETERMINE GREATER ACCURACY OF ANY DIAMETRIC DIMENSION, MEASURING OF CIRCUMFERENCE.IS REQUIRED
3. CONSTRUCTION per 49 CFR §178.504 (b)

MATERIAL:

1. DRUM BODY AND COVER CRCQ CARBON STEEL PER ASTM A1008 AND ASTM A568 OR EQUIVALENT
2. MATERIAL THICKNESS COVER 1.5mm (0.0543 - 0.0590; 16 GAUGE)
3. GASKET MAT'L PER ASTM D1056

FINISH:

1. DRUM BODY/BOTTOM EXTERIOR PAINTED BLACK (LQ10003) COVER EXTERIOR PAINTED WHITE (LQ10020)

ITEM	P/N	DESCRIPTION	QTY
7		UN DURABLE LABEL	1
6	NT5800	5/8-11UNC HEX JAM NUT	1
5	BT5840	5/8-11 X 4in LONG HEX HEAD BOLT	1
4	RBH2212	BOLT RING (HP) 55G 12GA 5/8 LUG	1
3	GE5500	GASKET D EPDM-BLACK	1
2	P2215	COVER 22.5 ID	1
1		BODY/BOTTOM 55 GALLON DRUM 22.5 ID	1

UN NUMBER EMBOSSED ON BOTTOM OF DRUM
½" CHARACTERS READING:
UN/1A2/X430/S/YR
UN/1A2/Y1.5/175/YR
1.5 *

(YR = YEAR OF MANUFACTURE)

TARE WEIGHT:
60 lbs

BEST PRINTED ON B-SIZE PAPER

THE USE OF INFORMATION CONTAINED HEREIN IS RESTRICTED FOR THE BENEFIT OF SKOLNIK INDUSTRIES ONLY. UNAUTHORIZED USE OR DISCLOSURE, COPYING OR OTHERWISE REPRODUCING (IN ANY MANNER) ANY PORTION OF THIS DRAWING WITHOUT PRIOR WRITTEN PERMISSION OF AN AUTHORIZED OFFICIAL OF SKOLNIK INDUSTRIES IS STRICTLY PROHIBITED.

MFG -	DRW LJT	CHK LJT
	SLS DR	QA GF
	APR MS	
	FILE CREATED IN AutoCAD 2002	

UNLESS STATED OTHERWISE
(DIMENSIONS IN PARENTHESIS ARE FOR REFERENCE ONLY!!!)
ALL DIMENSIONS ARE IN INCHES
DO NOT SCALE DRAWING
TOLERANCES:
FRACTIONS ±1/4
X° ± 3°
X.X ± 0.2
X.XX ± 0.10
X.XXX ± 0.030
3RD

SKOLNIK INDUSTRIES, INC.
4900 SOUTH KILBOURN AVENUE, CHICAGO, IL 60632

DESCRIPTION
55 GALLON OPEN HEAD DRUM
1.5 - 1.5 - 1.5 CRCQ
1A2/X430/S & 1A2/Y1.5/175

REL. DATE	PART NUM. / FILENAME	REV
07-NOV-2006	CQ5508	A5

Figure A-7. 55-Gallon Drum Detail Drawing (Skolnik Industries 2018).

A.2. Weak Hardening Curves

The weak hardening curves for the stainless and carbon steel were constructed by removing the strain hardening imparted during fabrication from the nominal, as-received, hardening curve calibrated in Reedlunn et al. (2019). Austenitic stainless steels, such as 304L, are strengthened principally via strain hardening, so removing the strain hardening effectively creates an annealed hardening curve. Removing the strain hardening is a less suitable method to generate an annealed carbon steel hardening curve, since carbon steel has several strengthening mechanisms, but both weak hardening curves were created by removing the strain hardening for simplicity. Note that the material model for both steels generates a continuous hardening curve by linearly interpolating between a series of user-specified equivalent plastic strain and equivalent hardening stress data points ($\bar{\varepsilon}^{\mathrm{pl}}$, H).

The strain hardening was removed from the nominal hardening curves using the following process. A polynomial was fit to each steel's nominal hardening curve, between $\bar{\varepsilon}^{\mathrm{pl}} = 0$ and 5 %. A third-order polynomial was used for the stainless steel, while a second-order polynomial was used for the carbon steel. Each polynomial was then extrapolated backwards to find the equivalent plastic strain $\bar{\varepsilon}^{\mathrm{pl}}_0$ that corresponded to the steel's minimum yield strength $\bar{\sigma}_{\mathrm{y0}}$. The minimum yield strengths were found in the appropriate ASTM standard. Table 4 of ASTM A312/A312M-18a (ASTM 2018a) lists a minimum yield strength of $\bar{\sigma}_{\mathrm{y0}} = 170$ MPa for TP 304L stainless steel, while Table 3 of ASTM A1008/A1008M-18 (ASTM 2018b) lists a minimum yield strength of $\bar{\sigma}_{\mathrm{y0}} = 140$ MPa for cold rolled commercial (carbon) steel sheet. To construct the weak hardening curve, $-\bar{\varepsilon}^{\mathrm{pl}}_0$ was added to each $\bar{\varepsilon}^{\mathrm{pl}}$ value in the nominal hardening curve and the point $(0, \bar{\sigma}_{\mathrm{y0}})$ was prepended to the resulting series of ($\bar{\varepsilon}^{\mathrm{pl}}$, H) points. The stainless and carbon steel weak hardening curve data points are listed in Table A-1. (See Table 2-3 and 2-6 in Reedlunn et al. (2019) for the nominal hardening curve data points.)

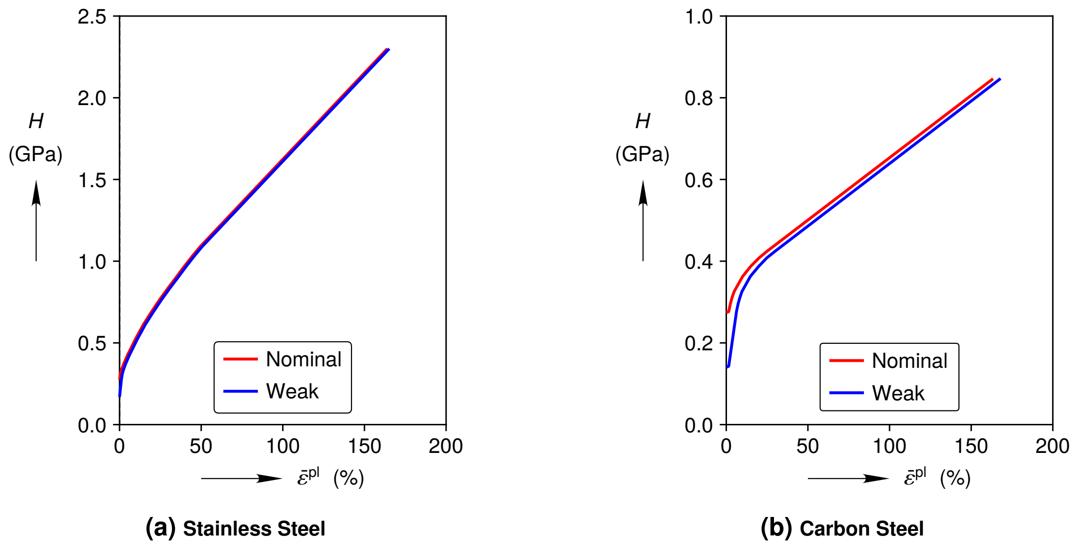


Figure A-8. Nominal and weak hardening curves.

The weak and nominal hardening data points are compared in Fig. A-8. Both weak hardening

Table A-1. Weak hardening curve data points

(a) Stainless Steel

$\bar{\varepsilon}^{\text{pl}}$ (unitless)	H (MPa)
0.000	170
0.012	277
0.016	308
0.023	331
0.035	366
0.060	423
0.109	523
0.159	614
0.209	692
0.258	765
0.308	834
0.358	900
0.407	966
0.457	1030
0.507	1090
1.652	2300

(b) Carbon Steel

$\bar{\varepsilon}^{\text{pl}}$ (unitless)	H (MPa)
0.000	140
0.015	143
0.063	277
0.073	296
0.083	310
0.093	322
0.098	327
0.148	363
0.198	387
0.248	408
0.298	424
1.680	847

curves are very similar to their nominal counterparts due to the rapid hardening between $\bar{\varepsilon}^{\text{pl}} = 0$ and 5 %. Aside from the data point corresponding to the minimum yield strength, each weak hardening curve is simply a shifted version of its nominal hardening curve, as expected.

A.3. Finite Element Meshes

The left side of Fig. A-9 depicts the finite element mesh of the geology utilized in the gradual compaction simulations. The magnified views on the right side of Fig. A-9 show the gradual compaction mesh near the room and the roof fall compaction mesh near the room. The roof fall block is shaded a darker grey to distinguish it from the surrounding, light grey, salt. The roof fall block is a separately meshed entity that interacts with the surrounding salt only through contact. Selective deviatoric hexahedral elements were used throughout the geomechanical models rather than Sierra/Solid Mechanics (2019)'s default uniform gradient hexahedral element with hourglass control. The element width and height near the room were $L_{\text{h}}/24$, while the thickness in the Y-direction was $L_{\text{d}}/5$ throughout the mesh. Cubit (Cubit Team 2017) journal files for both gradual compaction and roof fall scenarios can be found in Appendix A.4.1.

Figs. A-10a and A-10b show the 6-inch and 12-inch Standard container finite element meshes corresponding to the idealized container geometries in Fig. 2-2. Solid, selective deviatoric, elements represented the plywood, fiberboard, and pipe flanges, while Sierra/Solid Mechanics (2018)'s default, Belytschko-Tsay, shell elements represented the 55-gallon drums and pipes. Shell elements would have been more appropriate than solid elements for the thin layers of plywood, but we believe this choice did not significantly affect the results. Cubit (Cubit Team

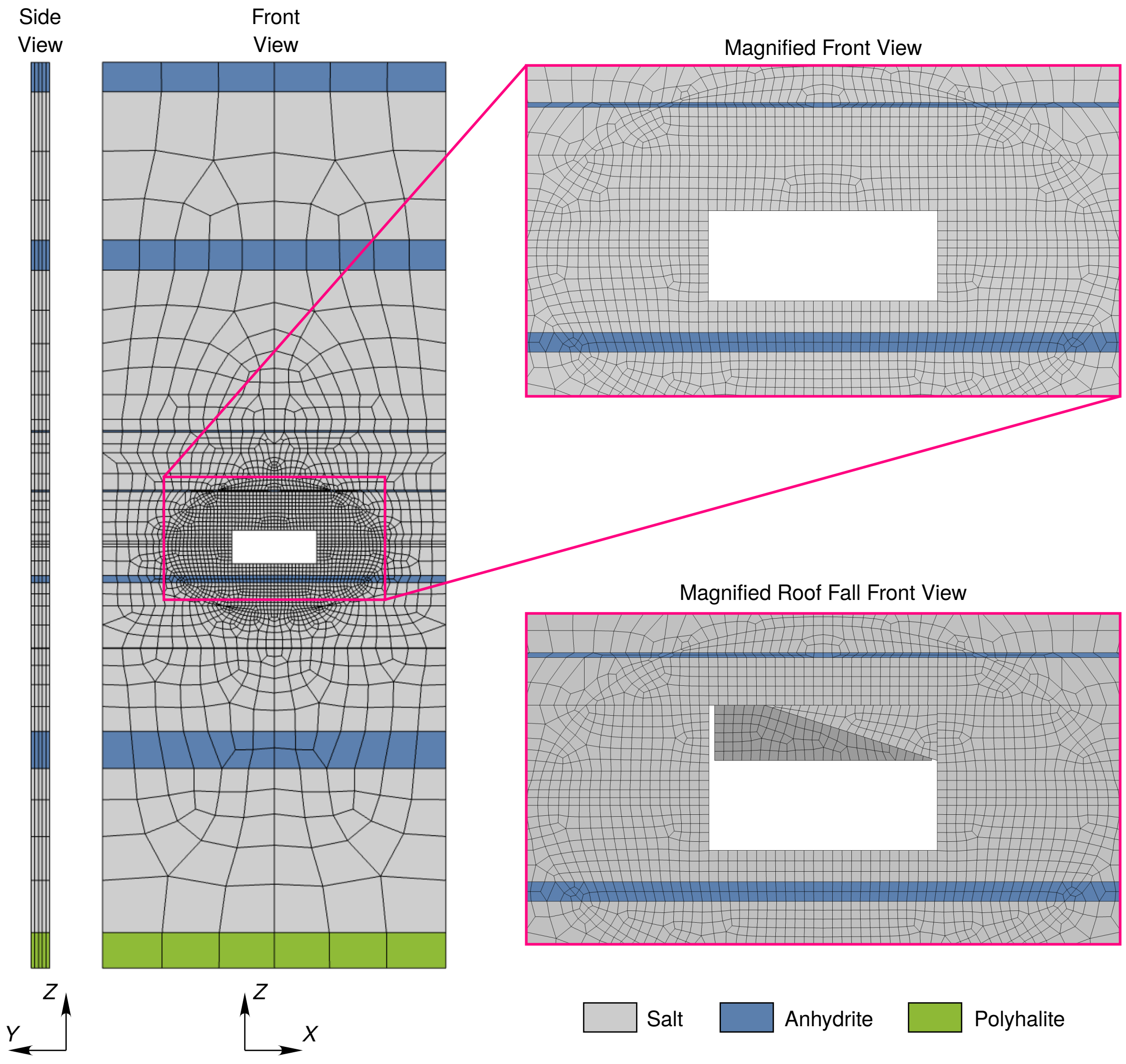
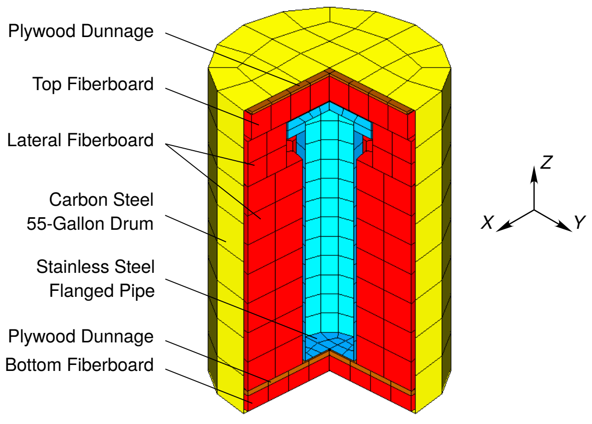


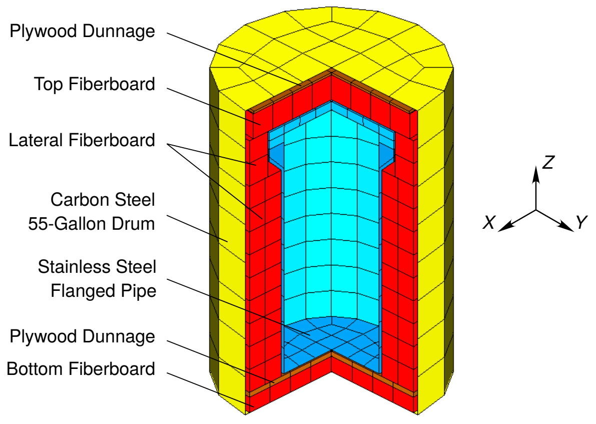
Figure A-9. Geology finite element meshes.

2017) journal files for the 6-inch and 12-inch Standard containers can be found in Appendix A.4.1.

The geology and container meshes were relatively coarse to attain reasonable simulation run times. Finer meshes would likely exhibit qualitatively similar results, yet with quantitatively greater container compaction. A mesh convergence study is left as potential future work.



(a) 6-inch Standard container



(b) 12-inch Standard container

Figure A-10. POP container finite element meshes. (Quarter wedge removed from meshes for visualization purposes.)

A.4. Simulation Files

A.4.1. Cubit Journal Files

6-inch Standard Container

```
# Project: Criticality of Pipe Overpack Container storage at WIPP Waste Disposal/Room D
#
# The following Pipe Overpack Container description is taken from-
#  Specification for Fabrication of the Pipe Overpack (Standard, S100, S200 and S300)
#  NWP Transportation Packaging Group Document Number:POC-SPC-0001, Rev. 2 May 2018

#  The Standard Pipe Overpack consists of a 6-inch (in.) or 12-in. pipe
#  component surrounded by fiberboard and plywood dunnage within a
#  standard 55-gallon drum with a rigid polyethylene liner and lid.
#  It is designed to be used for the shipment of specific
#  contact-handled transuranic waste forms in the TRUPACT-II and
#  the HalfPACT.

#  The Pipe Component is a stainless steel, cylindrical
#  pipe with a welded or formed bottom cap and a bolted stainless steel
#  lid sealed with a butyl rubber or ethylene propylene O-ring. The
#  pipe component is approximately 2 ft (feet) long and is available
#  with either a 6-in (0.280-in. nominal thickness) or a 12-in.
#  (0.250-in. nominal thickness) diameter. The pipe component must be
#  installed with a filter vent. The pipe component is centered in the
#  standard 55-gallon steel drum with dunnage consisting of fiberboard
#  packing and plywood.

#  The S100 Pipe Overpack consists of a 6-in. pipe
#  component surrounded by neutron shielding material on the sides and
#  by fiberboard and plywood dunnage on the top and bottom, within a
#  55-gallon drum with a rigid polyethylene liner and lid. In addition,
#  neutron shielding material is placed within the pipe component, above,
#  below, and around the payload. The S100 Pipe Overpack is intended for
#  the shipment of sealed neutron sources in the TRUPACT-II and the
#  HalfPACT.

#  The S200 Pipe Overpack consists of a gamma shield insert
#  located by rigid polyurethane foam dunnage inside a standard 12-in.
#  pipe component which is, in turn, located by fiberboard and plywood
#  dunnage within a standard 55 gallon drum with a rigid polyethylene
#  liner and lid. The 12-in. pipe component, fiberboard and plywood
#  dunnage, and 55-gallon drum with rigid polyethylene liner and lid are
#  identical to the Standard Pipe Overpack. The gamma shield insert is
#  a lead two-component assembly consisting of a cylindrical body with
#  an integral bottom cap and a detachable lid. The shield insert is
#  available in two sizes; the S200-A shield insert has a nominal
#  thickness of 1.000 in. and the S200-B shield insert has a nominal
#  thickness of 0.600 in. The overall dimensions of the S200-A and
#  S200-B shield inserts are nominally 10.125 in. diameter by 10.625 in.
#  long and 9.325 in. diameter by 17.825 in. long, respectively. The
#  rigid polyurethane foam dunnage fills the bottom and annular space
#  between the shield insert and the12-in. pipe component to position
#  the insert near the lid of the pipe component. The S200 Pipe Overpack
#  is intended for the shipment of transuranic waste forms with high gamma
#  energies in the TRUPACT-II and HalfPACT.

#  The S300 Pipe Overpack consists of a neutron shield insert placed inside
#  a standard 12-in. pipe component which is, in turn, located by fiberboard
#  and plywood dunnage within a standard 55- gallon drum with a rigid
#  polyethylene liner and lid. All of the components of the S300 Pipe
#  Overpack, except the neutron shield insert, are identical to the 12-in.
#  version of the Standard Pipe Overpack. The neutron shield insert is a
#  two-part assembly consisting of a cylindrical body and stepped lid. With
#  the exception of necessary clearances, the insert fits within and fills
#  the 12-in. pipe component. The insert lid is held in place by the lid
#  of the pipe component. The insert is made from solid, high-density
#  polyethylene (HDPE), and has a nominal side wall thickness of 4.13 in.
#  The S300 Pipe Overpack is intended for the shipment of sealed neutron
#  sources in the TRUPACT-II and HalfPACT.

# This journal file creates a simplified represertnation of a :
# Skolnik Industries 55 Gallon Open Head Drum (Part No. CQ5508)
#
# Created By: Jim Bean
# Using concepts developed by John Wilkes for the CCO container model
#
# Last Modified: {Version = "2019_07_31"}
#
# Software: CUBIT Version 15.3, Linux
#
# IMPORTANT:
#   CUBIT requires script playback be changed to the script directory
#   in order for output files to be in same directory as script file.
#
# Tools -> Options -> General -> Script Playback (Chg. to Script Dir)
#


Reset
#
view reset
rotate 135 about z
rotate -60 about x
#
#
# Selected Consistent System of Units
# Mass Len. Time Density Force Pres. Moment Temp. Energy Vel. Accel.
# kg   m    s    kg/m^3  N     Pa    N*m    K     J      m/s  m/s^2
#
#    Unit Conversion      #
# ft   = {ft_m = 0.3048}  m
# m    = {m_ft = 1/ft_m} ft
# in   = {in_m = 0.0254}  m
# m    = {m_in = 1/in_m} in
#                         #
```

```
#-----------------------------------------
#Begin description of the 55-Gallon drum components
#-----------------------------------------
# Definitions of Drum Geometry Variables
# Thickness(es) of Drum: {d_t = 1.50 / 1000} m
# Total Height of Drum: {d_h = 34.25 * in_m - d_t} m
# Diameter of Drum: {d_d = (22.50 * in_m) + d_t} m
# Radius of Drum: {d_r = d_d / 2} m
# Height of Round Seam: {d_hrs = 0.87 * in_m} m
#
### ------------- Drum Base ------------- ###
Create Surface Circle Radius {d_r}
Surface {Id("Surface")} Rename "S_D_B"
Volume {Id("Volume")} Rename "V_D_B"
surface with Name "S_D_B" Move Z {d_hrs + d_t/2.0}
#
### ------------- Drum Wall ------------- ###
Sweep Curve in Surface with Name "S_D_B" Vector 0 0 {d_h - 2.0*d_hrs}
Surface {Id("Surface")} Rename "S_D_W"
Volume {Id("Volume")} Rename "V_D_W"
Body {Id("Body")} Rename "B_D_W"
#
### ------------- Drum Lid ------------- ###
Surface with Name "S_D_B" Copy Move Z {d_h - 2.0*d_hrs}
Surface {Id("Surface")} Rename "S_D_L"
Volume {Id("Volume")} Rename "V_D_L"
Body {Id("Body")} Rename "B_D_L"
#
#--------------------------------------------
#Begin description of the dunnage components
#--------------------------------------------
# Define base fiberboard dunnage dimensions
# Bottom fiberboard spacer dimensions
# Ref. POC-DWG-0003 Figure 7
# Height of bottom fiberboard spacer   = {base_fib_h = 1.65  * in_m} m
# Diameter of bottom fiberboard spacer = {base_fib_d = 21.50 * in_m} m

#Create Fiberboard Base
Create Cylinder Height {base_fib_h} Radius {base_fib_d/ 2}
Surface {Id("Surface")} Rename "S_Base_Fib"
Volume {Id("Volume")} Rename "V_Base_Fib"
Body {Id("Body")} Rename "B_Base_Fib"
#{shift = d_hrs + d_t + (base_fib_h / 2)}
Body with Name "B_Base_Fib" Move Z {shift}

# Define base plywood dunnage dimensions
# Ref. POC-DWG-0003 Figure 8
# Height of bottom plywood spacer      = {base_ply_h = 0.5  * in_m} m
# Diameter of bottom plywood spacer    = {base_ply_d = 21.5 * in_m} m
#
#Create Plywood Base
Create Cylinder Height {base_ply_h} Radius {base_ply_d/ 2}
Surface {Id("Surface")} Rename "S_Base_Ply"
Volume {Id("Volume")} Rename "V_Base_Ply"
Body {Id("Body")} Rename "B_Base_Ply"
#{shift = d_hrs + d_t + base_fib_h + (base_ply_h / 2)}
Body with Name "B_Base_Ply" Move Z {shift}

#Define side wall fiberboard dimensions
# Ref. POC-DWG-0003 Figure 5
# Height of side fiberboard spacer     = {side_fib_h  = 21.9 * in_m} m
# Outer diameter of side spacer        = {side_fib_od = 21.5 * in_m} m
# Inner diameter of side spacer        = {side_fib_id = 7.0  * in_m} m
#
#Create side wall fiberboard dunnage
Create Cylinder Height {side_fib_h} Radius {side_fib_od/ 2}
Surface {Id("Surface")} Rename "S_Side_Fib"
```

```
Volume {Id("Volume")} Rename "V_Side_Fib"
Body {Id("Body")} Rename "B_Side_Fib"

# create cutting body for the dunnage wall
Create Cylinder Height {side_fib_h} Radius {side_fib_id/ 2}
Surface {Id("Surface")} Rename "S_Side_Fib_cut"
Volume {Id("Volume")} Rename "V_Side_Fib_cut"
Body {Id("Body")} Rename "B_Side_Fib_cut"
Subtract Volume with Name "V_Side_Fib_cut" from Volume with Name "V_Side_Fib"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + (side_fib_h / 2)}
Body with Name "B_Side_Fib" Move Z {shift}

#Define Upper Flange fiberboard spacer dimensions
# Ref. POC-DWG-0003 Fig. 4
# Height of flange fiberboard spacer   = {flange_fib_h  = 4.85 * in_m} m
# Outer diameter of flange spacer      = {flange_fib_od = 21.5 * in_m} m
# Inner diameter of flange spacer      = {flange_fib_id = 11.3 * in_m} m

#Create Upper Flange fiberboard spacer dunnage
Create Cylinder Height {flange_fib_h} Radius {flange_fib_od/ 2}
Surface {Id("Surface")} Rename "S_Flange_Fib"
Volume {Id("Volume")} Rename "V_Flange_Fib"
Body {Id("Body")} Rename "B_Flange_Fib"

# Create cutting body for the flange
Create Cylinder Height {flange_fib_h} Radius {flange_fib_id/ 2}
Surface {Id("Surface")} Rename "S_Flange_Fib_cut"
Volume {Id("Volume")} Rename "V_Flange_Fib_cut"
Body {Id("Body")} Rename "B_Flange_Fib_cut"

Subtract Volume with Name "V_Flange_Fib_cut" from Volume with Name "V_Flange_Fib"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + side_fib_h + (flange_fib_h / 2)}
Body with Name "B_Flange_Fib" Move Z {shift}

# Define Lid fiberboard spacer dimensions
# Ref. POC-DWG-0003 Fig. 2
# Height of lid fiberboard spacer      = {lid_fib_h = 2.65 * in_m} m
# Outer diameter of side spacer        = {lid_fib_d = 20.5 * in_m} m

#Create Lid fiberboard spacer
Create Cylinder Height {lid_fib_h} Radius {lid_fib_d/ 2}
Surface {Id("Surface")} Rename "S_Lid_Fib"
Volume {Id("Volume")} Rename "V_Lid_Fib"
Body {Id("Body")} Rename "B_Lid_Fib"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + side_fib_h + flange_fib_h + (lid_fib_h / 2)}
Body with Name "B_Lid_Fib" Move Z {shift}

# Define Lid Plywood Spacer Dimensions
# Ref. POC-DWG-0003 Fig. 3
# Height of lid plywood spacer         = {lid_ply_h = 0.25 * in_m} m
# Diameter of lid plywood spacer       = {lid_ply_d = 20.3 * in_m} m

#Create Lid Plywood Spacer
Create Cylinder Height {lid_ply_h} Radius {lid_ply_d/ 2}
Surface {Id("Surface")} Rename "S_Lid_Ply"
Volume {Id("Volume")} Rename "V_Lid_Ply"
Body {Id("Body")} Rename "B_Lid_Ply"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + side_fib_h + flange_fib_h +lid_fib_h + (lid_ply_h / 2)}
Body with Name "B_Lid_Ply" Move Z {shift}

#-------------------------------------------
# Begin description of the Pipe components
#------------------------------------------
### ------------- Pipe Bottom Cap and wall- using shell elements for these parts
# Ref. POC-DWG-0002 Section G-G
# Thickness of pipe bottom cap = {cap_thick = 0.28 * in_m} m
# Thickness of pipe wall = {wall_thick = 0.28 * in_m} m
# Ref. POC-DWG-0002 Section C-C
# Outer diameter of pipe bottom cap = {cap_od = 6.70 * in_m} m
# Length of wall = {wall_len = 25.625 * in_m} m
#
# Pipe Bottom- shell
Create Surface Circle Radius {(cap_od - wall_thick)/2 }
Surface {Id("Surface")} Rename "S_Pipe_Bot"
Volume {Id("Volume")} Rename "V_Pipe_Bot"
#
#Pipe Wall- shell
# The commented out line looks like it is correct but probably doesn't make any difference
#Sweep Curve in Surface with Name "S_Pipe_Bot" Vector 0 0 {wall_len - 0.5 * cap_thick}
Sweep Curve in Surface with Name "S_Pipe_Bot" Vector 0 0 {wall_len}
Surface {Id("Surface")} Rename "S_Pipe_Wall"
Volume {Id("Volume")} Rename "V_Pipe_Wall"
Body {Id("Body")} Rename "B_Pipe_Wall"
#
### ------------- Pipe Flange and lid- using continuum elements for these parts
# ignoring bolts. Combining the pipe flange and lid into 1 part implies the
# bolts don't fail
# Ref. POC-DWG-0002 Detail 3 and Section D-D
# Thickness of pipe flange = {flange_thick = 0.9 * in_m} m
# Thickness of pipe lid    = {lid_thick    = 0.9 * in_m} m
# Flange and lid diameter  = {lid_dia      = 11.03 * in_m} m

Create Cylinder Height {flange_thick + lid_thick} Radius {lid_dia / 2}
Surface {Id("Surface")} Rename "S_Pipe_Lid"
Volume {Id("Volume")} Rename "V_Pipe_Lid"
Body {Id("Body")} Rename "B_Pipe_Lid"

# Create cutting body
Create Cylinder Height {flange_thick} Radius {(cap_od - wall_thick)/2}
Surface {Id("Surface")} Rename "S_Pipe_cut"
Volume {Id("Volume")} Rename "V_Pipe_cut"
Body {Id("Body")} Rename "B_Pipe_cut"
Body with Name "B_Pipe_cut" Move Z {-0.5*flange_thick}
Subtract Volume with Name "V_Pipe_cut" from Volume with Name "V_Pipe_Lid"
Body with Name "B_Pipe_Lid" Move Z {wall_len}

Body with name    "B_Pipe_Lid"  Move Z {d_hrs + d_t + base_fib_h + base_ply_h + 0.5 * cap_thick}
Body with name    "B_Pipe_Wall" Move Z {d_hrs + d_t + base_fib_h + base_ply_h + 0.5 * cap_thick}
Surface with Name "S_Pipe_Bot"  Move Z {d_hrs + d_t + base_fib_h + base_ply_h + 0.5 * cap_thick}


###   ==================   QUARTER SYMMETRY  ==================   ###
#
Webcut Volume All with Plane yplane center 0
Webcut Volume All with Plane xplane center 0
#
Delete Volume with Name "*@*"
#Imprint and merge Pipe parts
imprint volume with Name "V_Pipe_Lid" with volume with Name "V_Pipe_Wall"
merge   volume with Name "V_Pipe_Lid" with volume with name "V_Pipe_Wall"
imprint volume with Name "V_Pipe_Bot" with volume with Name "V_Pipe_Wall"
merge   volume with Name "V_Pipe_Bot" with volume with name "V_Pipe_Wall"

#Imprint and merge Drum parts
imprint volume with Name "V_D_L" with volume with Name "V_D_W"
merge volume with Name "V_D_L" with volume with Name "V_D_W"
imprint volume with Name "V_D_B" with volume with Name "V_D_W"
merge volume with Name "V_D_B" with volume with Name "V_D_W"

###    ====================     MESH      ====================    ###
#
##   ~~~~~~~~~~~~~~~~~~~    55-Gallon Drum   ~~~~~~~~~~~~~~~~~~~   ##
#{mesh_factor = 0.35}
merge volume with name "V_D*"
volume with name "V_D*" size {d_h/(25 * mesh_factor)}
mesh surface in volume with name "V_D*"
```

```
#
##   ~~~~~~~~~~~~~~~~~~~    Dunnage    ~~~~~~~~~~~~~~~~~~~   ##
# Fiberboard and Plywood
merge volume with name "V_B*"
volume with name "V_B*" size {base_ply_d/(25 * mesh_factor)}
mesh volume with name "V_B*"
#
# Side Fiberboard
merge volume with name "V_S*"
volume with name "V_S*" size {0.5*(side_fib_od - side_fib_id)/(7 * mesh_factor)}
mesh volume with name "V_S*"

# Flange Fiberboard
merge volume with name "V_F*"
volume with name "V_F*" size {0.5*(flange_fib_od - flange_fib_id)/(5 * mesh_factor)}
mesh volume with name "V_F*

# Lid Fiberboard and Plywood Spacer
merge volume with name "V_Lid*"
volume with name "V_Lid*" size {lid_ply_d/(25 * mesh_factor)}
mesh volume with name "V_Lid*

##   ------------- Pipe ------------- ###
# Merge All Portions on Pipe with their respective adjoining flanges
Merge Volume with Name "V_P*"
volume with name "V_P*" size {flange_fib_h/(7 * mesh_factor)}
# Mesh the pipe lid
mesh volume with name "V_Pipe_Lid*"
# Mesh the pipe wall and bottom Surfaces
mesh surface in volume with name "V_Pipe_Wall"
mesh surface in volume with name "V_Pipe_Bot"

#
#
###   ===   CONVERT QUARTER SYMMETRY MODEL TO A FULL MODEL  ===   ###
Volume All Copy Reflect X
Volume All Copy Reflect Y
Merge Volume with Name "V_D*"
Merge Volume with Name "V_B*"
Merge Volume with Name "V_F*"
Merge Volume with Name "V_L*"
Merge Volume with Name "V_P*"
Merge Volume with Name "V_S*"
#
#
#
###  === Create Blocks  ===  ###
block 100 surface in volume with name "V_D_W*"
block 100 name 'pop_drum_wall'
#
block 101 surface in volume with name "V_D_B*"
block 101 name 'pop_drum_base'
#
block 102 surface in volume with name "V_D_L*"
block 102 name 'pop_drum_lid'
#
block 200 volume with name "V_Base_Ply*"
block 200 name 'pop_plywood_lower'
#
block 201 volume with name "V_Lid_Ply*"
block 201 name 'pop_plywood_upper'
#
block 202 volume with name "V_Base_Fib*"
block 202 name 'pop_fiberboard_lower'
#
block 203 volume with name "V_Lid_Fib*"
block 203 name 'pop_fiberboard_upper'
#
```

```
block 204 volume with name "V_Side_Fib*"
block 204 name 'pop_fiberboard_side'

block 205 volume with name "V_Flange_Fib*"
block 205 name 'pop_fiberboard_flange

block 300 surface in volume with name "V_Pipe_W*"
block 300 name 'pop_cv_wall'
#
block 301 surface in volume with name "V_Pipe_B*"
block 301 name 'pop_cv_base'
#
block 302 volume with name "V_Pipe_L*"
block 302 name 'pop_cv_top_flange'
#
# Move assembly so that it is centered at 0 0 0
group 'pop_assembly' add volume all
move group with name 'pop_assembly' midpoint location 0.0 0.0 0.0

nodeset 1  add surface with name "S_Pipe_Wall*"
nodeset 1 name 'tracker_ns_1'

#
#
###  ===  Export Mesh  ===  ###
#{_file_name = "'single_6inch_pop" // ".g'"}
export mesh {_file_name} dimension 3 overwrite

# Save .cub file
#{_file_name = "'single_6inch_pop" // ".cub'"}
save as {_file_name} overwrite
```

12-inch Standard Container

```
# Project: Criticality of Pipe Overpack Container storage at WIPP Waste Disposal/Room D
#
# The following Pipe Overpack Container description is taken from-
#  Specification for Fabrication of the Pipe Overpack (Standard, S100, S200 and S300)
#  NWP Transportation Packaging Group Document Number:POC-SPC-0001, Rev. 2 May 2018

#  The Standard Pipe Overpack consists of a 6-inch (in.) or 12-in. pipe
#  component surrounded by fiberboard and plywood dunnage within a
#  standard 55-gallon drum with a rigid polyethylene liner and lid.
#  It is designed to be used for the shipment of specific
#  contact-handled transuranic waste forms in the TRUPACT-II and
#  the HalfPACT.

#  The Pipe Component is a stainless steel, cylindrical
#  pipe with a welded or formed bottom cap and a bolted stainless steel
#  lid sealed with a butyl rubber or ethylene propylene O-ring. The
#  pipe component is approximately 2 ft (feet) long and is available
#  with either a 6-in (0.280-in. nominal thickness) or a 12-in.
#  (0.250-in. nominal thickness) diameter. The pipe component must be
#  installed with a filter vent. The pipe component is centered in the
#  standard 55-gallon steel drum with dunnage consisting of fiberboard
#  packing and plywood.

#  The S100 Pipe Overpack consists of a 6-in. pipe
#  component surrounded by neutron shielding material on the sides and
#  by fiberboard and plywood dunnage on the top and bottom, within a
#  55-gallon drum with a rigid polyethylene liner and lid. In addition,
#  neutron shielding material is placed within the pipe component, above,
#  below, and around the payload. The S100 Pipe Overpack is intended for
#  the shipment of sealed neutron sources in the TRUPACT-II and the
#  HalfPACT.
#  The S200 Pipe Overpack consists of a gamma shield insert
#  located by rigid polyurethane foam dunnage inside a standard 12-in.
#  pipe component which is, in turn, located by fiberboard and plywood
#  dunnage within a standard 55 gallon drum with a rigid polyethylene
#  liner and lid. The 12-in. pipe component, fiberboard and plywood
#  dunnage, and 55-gallon drum with rigid polyethylene liner and lid are
#  identical to the Standard Pipe Overpack. The gamma shield insert is
#  a lead two-component assembly consisting of a cylindrical body with
#  an integral bottom cap and a detachable lid. The shield insert is
#  available in two sizes; the S200-A shield insert has a nominal
#  thickness of 1.000 in. and the S200-B shield insert has a nominal
#  thickness of 0.600 in. The overall dimensions of the S200-A and
#  S200-B shield inserts are nominally 10.125 in. diameter by 10.625 in.
#  long and 9.325 in. diameter by 17.825 in. long, respectively. The
#  rigid polyurethane foam dunnage fills the bottom and annular space
#  between the shield insert and the12-in. pipe component to position
#  the insert near the lid of the pipe component. The S200 Pipe Overpack
#  is intended for the shipment of transuranic waste forms with high gamma
#  energies in the TRUPACT-II and HalfPACT.

#  The S300 Pipe Overpack consists of a neutron shield insert placed inside
#  a standard 12-in. pipe component which is, in turn, located by fiberboard
#  and plywood dunnage within a standard 55- gallon drum with a rigid
#  polyethylene liner and lid. All of the components of the S300 Pipe
#  Overpack, except the neutron shield insert, are identical to the 12-in.
#  version of the Standard Pipe Overpack. The neutron shield insert is a
#  two-part assembly consisting of a cylindrical body and stepped lid. With
#  the exception of necessary clearances, the insert fits within and fills
#  the 12-in. pipe component. The insert lid is held in place by the lid
#  of the pipe component. The insert is made from solid, high-density
#  polyethylene (HDPE), and has a nominal side wall thickness of 4.13 in.
#  The S300 Pipe Overpack is intended for the shipment of sealed neutron
#  sources in the TRUPACT-II and HalfPACT.

# This journal file creates a simplified represertnation of a :
# Skolnik Industries 55 Gallon Open Head Drum (Part No. CQ5508)
#
# Created By: Jim Bean
# Using concepts developed by John Wilkes for the CCO container model
#
# Last Modified: {Version = "2019_01_19"}
#
# Software: CUBIT Version 15.3, Linux
#
# IMPORTANT:
#   CUBIT requires script playback be changed to the script directory
#   in order for output files to be in same directory as script file.
#
# Tools -> Options -> General -> Script Playback (Chg. to Script Dir)
#


Reset
#
view reset
rotate 135 about z
rotate -60 about x
#
#
# Selected Consistent System of Units
# Mass Len. Time Density Force Pres. Moment Temp. Energy Vel. Accel.
# kg   m    s    kg/m^3  N     Pa    N*m    K     J      m/s  m/s^2
#
#    Unit Conversion      #
# ft   = {ft_m = 0.3048}  m
# m    = {m_ft = 1/ft_m} ft
# in   = {in_m = 0.0254}  m
# m    = {m_in = 1/in_m} in
#                        #

#-----------------------------------------
#Begin description of the 55-Gallon drum components
#-----------------------------------------
# Definitions of Drum Geometry Variables
# Thickness(es) of Drum: {d_t = 1.50 / 1000} m
# Total Height of Drum: {d_h = 34.25 * in_m - d_t} m
# Diameter of Drum: {d_d = (22.50 * in_m) + d_t} m
# Radius of Drum: {d_r = d_d / 2} m
# Height of Round Seam: {d_hrs = 0.87 * in_m} m
#
### ------------- Drum Base ------------- ###
Create Surface Circle Radius {d_r}
Surface {Id("Surface")} Rename "S_D_B"
Volume {Id("Volume")} Rename "V_D_B"
surface with Name "S_D_B" Move Z {d_hrs + d_t/2.0}
#
### ------------- Drum Wall ------------- ###
Sweep Curve in Surface with Name "S_D_B" Vector 0 0 {d_h - 2.0*d_hrs}
Surface {Id("Surface")} Rename "S_D_W"
Volume {Id("Volume")} Rename "V_D_W"
Body {Id("Body")} Rename "B_D_W"
#
### ------------- Drum Lid ------------- ###
Surface with Name "S_D_B" Copy Move Z {d_h - 2.0*d_hrs}
Surface {Id("Surface")} Rename "S_D_L"
Volume {Id("Volume")} Rename "V_D_L"
Body {Id("Body")} Rename "B_D_L"
#
#--------------------------------------------
#Begin description of the dunnage components
#--------------------------------------------
# Define base fiberboard dunnage dimensions
# Bottom fiberboard spacer dimensions
# Ref. POC-DWG-0007 Fig. 7
# Height of bottom fiberboard spacer   = {base_fib_h = 1.65  * in_m} m
# Diameter of bottom fiberboard spacer = {base_fib_d = 21.50 * in_m} m

#Create Fiberboard Base
Create Cylinder Height {base_fib_h} Radius {base_fib_d/ 2}
Surface {Id("Surface")} Rename "S_Base_Fib"
Volume {Id("Volume")} Rename "V_Base_Fib"
Body {Id("Body")} Rename "B_Base_Fib"
#{shift = d_hrs + d_t + (base_fib_h / 2)}
Body with Name "B_Base_Fib" Move Z {shift}

# Define base plywood dunnage dimensions
# Ref. POC-DWG-0007 Fig. 8
# Height of bottom plywood spacer      = {base_ply_h = 0.5  * in_m} m
# Diameter of bottom plywood spacer    = {base_ply_d = 21.5 * in_m} m
#
#Create Plywood Base
Create Cylinder Height {base_ply_h} Radius {base_ply_d/ 2}
Surface {Id("Surface")} Rename "S_Base_Ply"
Volume {Id("Volume")} Rename "V_Base_Ply"
Body {Id("Body")} Rename "B_Base_Ply"
#{shift = d_hrs + d_t + base_fib_h + (base_ply_h / 2)}
Body with Name "B_Base_Ply" Move Z {shift}

#Define side wall fiberboard dimensions
# Ref. POC-DWG-0007 Fig. 5
# Height of side fiberboard spacer     = {side_fib_h  = 22.0 * in_m} m
# Outer diameter of side spacer        = {side_fib_od = 21.5 * in_m} m
# Inner diameter of side spacer        = {side_fib_id = 13.1 * in_m} m
#
#Create side wall fiberboard dunnage
```

```
Create Cylinder Height {side_fib_h} Radius {side_fib_od/ 2}
Surface {Id("Surface")} Rename "S_Side_Fib"
Volume {Id("Volume")} Rename "V_Side_Fib"
Body {Id("Body")} Rename "B_Side_Fib"

# create cutting body for the dunnage wall
Create Cylinder Height {side_fib_h} Radius {side_fib_id/ 2}
Surface {Id("Surface")} Rename "S_Side_Fib_cut"
Volume {Id("Volume")} Rename "V_Side_Fib_cut"
Body {Id("Body")} Rename "B_Side_Fib_cut"
Subtract Volume with Name "V_Side_Fib_cut" from Volume with Name "V_Side_Fib"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + (side_fib_h / 2)}
Body with Name "B_Side_Fib" Move Z {shift}

#Define Upper Flange fiberboard spacer dimensions
# Ref. POC-DWG-0007 Fig. 4
# Height of flange fiberboard spacer   = {flange_fib_h  = 4.85 * in_m} m
# Outer diameter of flange spacer      = {flange_fib_od = 21.5 * in_m} m
# Inner diameter of flange spacer      = {flange_fib_id = 16.6 * in_m} m

#Create Upper Flange fiberboard spacer dunnage
Create Cylinder Height {flange_fib_h} Radius {flange_fib_od/ 2}
Surface {Id("Surface")} Rename "S_Flange_Fib"
Volume {Id("Volume")} Rename "V_Flange_Fib"
Body {Id("Body")} Rename "B_Flange_Fib"

# Create cutting body for the flange
Create Cylinder Height {flange_fib_h} Radius {flange_fib_id/ 2}
Surface {Id("Surface")} Rename "S_Flange_Fib_cut"
Volume {Id("Volume")} Rename "V_Flange_Fib_cut"
Body {Id("Body")} Rename "B_Flange_Fib_cut"

Subtract Volume with Name "V_Flange_Fib_cut" from Volume with Name "V_Flange_Fib"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + side_fib_h + (flange_fib_h / 2)}
Body with Name "B_Flange_Fib" Move Z {shift}

# Define Lid fiberboard spacer dimensions
# Ref. POC-DWG-0007 Fig. 2
# Height of lid fiberboard spacer      = {lid_fib_h = 2.65 * in_m} m
# Outer diameter of side spacer        = {lid_fib_d = 20.5 * in_m} m

#Create Lid fiberboard spacer
Create Cylinder Height {lid_fib_h} Radius {lid_fib_d/ 2}
Surface {Id("Surface")} Rename "S_Lid_Fib"
Volume {Id("Volume")} Rename "V_Lid_Fib"
Body {Id("Body")} Rename "B_Lid_Fib"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + side_fib_h + flange_fib_h + (lid_fib_h / 2)}
Body with Name "B_Lid_Fib" Move Z {shift}

# Define Lid Plywood Spacer Dimensions
# Ref. POC-DWG-0007 Fig. 3
# Height of lid plywood spacer         = {lid_ply_h = 0.25 * in_m} m
# Diameter of lid plywood spacer       = {lid_ply_d = 20.3 * in_m} m

#Create Lid Plywood Spacer
Create Cylinder Height {lid_ply_h} Radius {lid_ply_d/ 2}
Surface {Id("Surface")} Rename "S_Lid_Ply"
Volume {Id("Volume")} Rename "V_Lid_Ply"
Body {Id("Body")} Rename "B_Lid_Ply"
#{shift = d_hrs + d_t + base_fib_h + base_ply_h + side_fib_h + flange_fib_h +lid_fib_h + (lid_ply_h / 2)}
Body with Name "B_Lid_Ply" Move Z {shift}

#-------------------------------------------
# Begin description of the Pipe components
#------------------------------------------
### ------------- Pipe Bottom Cap and wall- using shell elements for these parts
# Ref. POC-DWG-0006 Section G-G
# Thickness of pipe bottom cap = {cap_thick = 0.28 * in_m} m
# Thickness of pipe wall = {wall_thick = 0.25 * in_m} m
# Outer diameter of pipe bottom cap = {cap_od = 12.75 * in_m} m
# Ref. POC-DWG-0007 Section C-C
# Length of wall = {wall_len = 25.625 * in_m} m
#
# Pipe Bottom- shell
Create Surface Circle Radius {(cap_od - wall_thick)/2 }
Surface {Id("Surface")} Rename "S_Pipe_Bot"
Volume {Id("Volume")} Rename "V_Pipe_Bot"
#
#Pipe Wall- shell
# The commented out line looks like it is correct but probably doesn't make any difference
#Sweep Curve in Surface with Name "S_Pipe_Bot" Vector 0 0 {wall_len - 0.5 * cap_thick}
Sweep Curve in Surface with Name "S_Pipe_Bot" Vector 0 0 {wall_len}
Surface {Id("Surface")} Rename "S_Pipe_Wall"
Volume {Id("Volume")} Rename "V_Pipe_Wall"
Body {Id("Body")} Rename "B_Pipe_Wall"
#
### ------------- Pipe Flange and lid- using continuum elements for these parts
# ignoring bolts. Combining the pipe flange and lid into 1 part implies the
# bolts don't fail
# Ref. POC-DWG-0006 Detail 3 and Section D-D
# Thickness of pipe flange = {flange_thick = 0.9 * in_m} m
# Thickness of pipe lid    = {lid_thick    = 0.9 * in_m} m
# Flange and lid diameter  = {lid_dia      = 16.35 * in_m} m

Create Cylinder Height {flange_thick + lid_thick} Radius {lid_dia / 2}
Surface {Id("Surface")} Rename "S_Pipe_Lid"
Volume {Id("Volume")} Rename "V_Pipe_Lid"
Body {Id("Body")} Rename "B_Pipe_Lid"

# Create cutting body
Create Cylinder Height {flange_thick} Radius {(cap_od - wall_thick)/2}
Surface {Id("Surface")} Rename "S_Pipe_cut"
Volume {Id("Volume")} Rename "V_Pipe_cut"
Body {Id("Body")} Rename "B_Pipe_cut"
Body with Name "B_Pipe_cut" Move Z {-0.5*flange_thick}
Subtract Volume with Name "V_Pipe_cut" from Volume with Name "V_Pipe_Lid"
Body with Name "B_Pipe_Lid" Move Z {wall_len}

Body with name    "B_Pipe_Lid"  Move Z {d_hrs + d_t + base_fib_h + base_ply_h + 0.5 * cap_thick}
Body with name    "B_Pipe_Wall" Move Z {d_hrs + d_t + base_fib_h + base_ply_h + 0.5 * cap_thick}
Surface with Name "S_Pipe_Bot"  Move Z {d_hrs + d_t + base_fib_h + base_ply_h + 0.5 * cap_thick}


###   ==================   QUARTER SYMMETRY  ==================   ###
#
Webcut Volume All with Plane yplane center 0
Webcut Volume All with Plane xplane center 0
#
Delete Volume with Name "*@*"
#Imprint and merge Pipe parts
imprint volume with Name "V_Pipe_Lid" with volume with Name "V_Pipe_Wall"
merge   volume with Name "V_Pipe_Lid" with volume with name "V_Pipe_Wall"
imprint volume with Name "V_Pipe_Bot" with volume with Name "V_Pipe_Wall"
merge   volume with Name "V_Pipe_Bot" with volume with name "V_Pipe_Wall"

#Imprint and merge Drum parts
imprint volume with Name "V_D_L" with volume with Name "V_D_W"
merge volume with Name "V_D_L" with volume with Name "V_D_W"
imprint volume with Name "V_D_B" with volume with Name "V_D_W"
merge volume with Name "V_D_B" with volume with Name "V_D_W"

###    ====================     MESH      ====================    ###
#
##   ~~~~~~~~~~~~~~~~~~~    55-Gallon Drum   ~~~~~~~~~~~~~~~~~~~   ##
#{mesh_factor = 0.35}
merge volume with name "V_D*"
```

```
volume with name "V_D*" size {d_h/(25 * mesh_factor)}
mesh surface in volume with name "V_D*"
#
##   ~~~~~~~~~~~~~~~~~~~    Dunnage    ~~~~~~~~~~~~~~~~~~~   ##
# Fiberboard and Plywood
merge volume with name "V_B*"
volume with name "V_B*" size {base_ply_d/(25 * mesh_factor)}
mesh volume with name "V_B*"
#
# Side Fiberboard
merge volume with name "V_S*"
volume with name "V_S*" size {0.5*(side_fib_od - side_fib_id)/(5 * mesh_factor)}
mesh volume with name "V_S*"

# Flange Fiberboard
merge volume with name "V_F*"
volume with name "V_F*" size {0.5*(flange_fib_od - flange_fib_id)/(3 * mesh_factor)}
mesh volume with name "V_F*

# Lid Fiberboard and Plywood Spacer
merge volume with name "V_Lid*"
volume with name "V_Lid*" size {lid_ply_d/(25 * mesh_factor)}
mesh volume with name "V_Lid*

##   ------------- Pipe ------------- ###
# Merge All Portions on Pipe with their respective adjoining flanges
Merge Volume with Name "V_P*"
volume with name "V_P*" size {flange_fib_h/(5 * mesh_factor)}
# Mesh the pipe lid
mesh volume with name "V_Pipe_Lid*"
# Mesh the pipe wall and bottom Surfaces
mesh surface in volume with name "V_Pipe_Wall"
mesh surface in volume with name "V_Pipe_Bot"

#
#
###   ===   CONVERT QUARTER SYMMETRY MODEL TO A FULL MODEL  ===   ###
Volume All Copy Reflect X
Volume All Copy Reflect Y
Merge Volume with Name "V_D*"
Merge Volume with Name "V_B*"
Merge Volume with Name "V_F*"
Merge Volume with Name "V_L*"
Merge Volume with Name "V_P*"
Merge Volume with Name "V_S*"
#
#
#
###  === Create Blocks  ===  ###
block 100 surface in volume with name "V_D_W*"
block 100 name 'pop_drum_wall'
#
block 101 surface in volume with name "V_D_B*"
block 101 name 'pop_drum_base'
#
block 102 surface in volume with name "V_D_L*"
block 102 name 'pop_drum_lid'
#
block 200 volume with name "V_Base_Ply*"
block 200 name 'pop_plywood_lower'
#
block 201 volume with name "V_Lid_Ply*"
block 201 name 'pop_plywood_upper'
#
block 202 volume with name "V_Base_Fib*"
block 202 name 'pop_fiberboard_lower'
#
block 203 volume with name "V_Lid_Fib*"
```

```
block 203 name 'pop_fiberboard_upper'
#
block 204 volume with name "V_Side_Fib*"
block 204 name 'pop_fiberboard_side'

block 205 volume with name "V_Flange_Fib*"
block 205 name 'pop_fiberboard_flange

block 300 surface in volume with name "V_Pipe_W*"
block 300 name 'pop_cv_wall'
#
block 301 surface in volume with name "V_Pipe_B*"
block 301 name 'pop_cv_base'
#
block 302 volume with name "V_Pipe_L*"
block 302 name 'pop_cv_top_flange'
#
# Move assembly so that it is centered at 0 0 0
group 'pop_assembly' add volume all
move group with name 'pop_assembly' midpoint location 0.0 0.0 0.0

nodeset 1  add surface with name "S_Pipe_Wall*"
nodeset 1 name 'tracker_ns_1'

#
#
###  ===  Export Mesh  ===  ###
#{_file_name = "'detailed_pop_" // Version // ".g'"}
export mesh {_file_name} dimension 3 overwrite

# Save .cub file
#{_file_name = "'detailed_pop_" // Version // ".cub'"}
save as {_file_name} overwrite
```

Gradual Compaction, Disposal Room

```
#!python
#Cubit v15.0, and v15.3 (v15.1 and v15.2 don't always mesh the surfaces)
cubit.cmd('reset')
cubit.set_playback_paused_on_error(False)
error_old = cubit.get_error_count()
cubit.cmd('undo off')
import math
from collections import OrderedDict
import re

############################################
#  WIPP lower horizon geologic mesh creation
############################################

# Define conversion factors
in_m = 0.0254

#mesh_name = 'disposal_room-no_clay_slip'
mesh_name = 'disposal_room-with_clay_slip-lower_horizon'

#Initialize dictionary to store geometry values
#(All dimensions are in meters.)
g = dict()

#Define whether to model clay seams as slipping surfaces
#clay_seam_slip = False
clay_seam_slip = True
#Define whether to split each element into 4 elements
split_elements = False
```

```
#Define the clay seam slip definition
#seam_slip_defn = 'SAND89-2948'
seam_slip_defn = 'all'
#Define whether to create blocks for the layer of elements on the interior of room
room_surf_blocks = False
#Define whether to assume symmetry across the X=0 plane
#x_sym = True
x_sym = False

#Define the room dimensions as defined in SAND97-0795
#(elevations relative to clay seam G)
#(Figure 3 in SAND97-0795 has the room top elevation as -1.96 m and the bottom as -5.92 m,
#while Figure 4 has the room top elevation at -2.43 m and the bottom at -6.39 m.  This discrepancy
#is discussed in a March 4 memo from Charles Stone to B. Butcher, which is in Appendix B-8 of
#SAND97-0796.  Apparently they decided to move the room elevation because they felt it was most
#important to have the distance between marker band 139 and the disposal room floor be 1.39 m.)
g['room_top_elev'] = -2.43
g['room_bot_elev'] = -6.39
g['room_width'] = 5.03 * 2.0

#(This fillet radius is from the cutting head radius used in room D, plus a small amount.  I do not
#know if the disposal rooms used the same cutting head as Room D, but it seems like a safe bet.)
#g['room_fillet_radius'] = 0.00001 # 18.0 * 25.4 / 1000.0

#JEB- using Cubit 15-4 with the original room_fillet_radius causes some elements at the corner of the room
#to have nearly coincident nodes which results in Sierra/SM computing a critical time step O(10^-9) sec.
#Setting the radius to zero eliminates the problem.
g['room_fillet_radius'] =0.0

#Define the coordinates of the center of the room
g['room_ctr'] = [0.0, (g['room_bot_elev'] + g['room_top_elev']) / 2.0]

#Define the total simulation dimensions as defined in SAND97-0795
#(elevations relative to the bottom of the bottom halite layer)
g['top_elev'] = 52.87
g['bot_elev'] = -54.19
g['width'] = 20.27 * 2.0
CCO_gap = 5.0 / 1000.0
#CCO_dia = 24.0 * 25.4 / 1000.0 + CCO_gap
d_t = 1.50 / 1000
CCO_dia = (22.50 * in_m) + 2.0*d_t # JLB Detailed CCO
seven_pack_depth = CCO_dia + (CCO_dia + CCO_gap) * math.cos(math.pi/6) * 2.0 + CCO_gap  # JLB - Added CCO_gap
num_CCO_rows = 1.0
CCO_row_depth = seven_pack_depth + CCO_dia # add extra drum diameter
g['thickness'] = CCO_row_depth * num_CCO_rows

#Define the fine element size in the X-Z plane
H_xz = g['room_width'] / 24.0
#Define the fine element size in the Y-direction
H_y = H_xz * 1.0
print H_y
#Define the radius of the fine region
R_fine = [g['room_width'] * 1.25, (g['room_top_elev'] - g['room_bot_elev']) * 2]
#Define the coarsening rate
coarsen_rate = 0.36

#Define stratigraphy
#
#(bot is the bottom of each layer, relative to the bottom of the halite in the room.
#Clay defines whether there is a sliding layer of clay at the bottom.)
#
#(These definitions come from SAND97-0795)
#layers = OrderedDict( [\
#    ('arg_halite_1', dict(bot = -54.19, clay = None)), \
#    ('anhydrite_1', dict(bot = -8.63, clay = 'E')), \
#    ('arg_halite_2', dict(bot = -7.77, clay = None)), \
#    ('halite_1', dict(bot = 0.0, clay = 'G')), \
#    ('anhydrite_2', dict(bot = 2.10, clay = 'H')), \
#     ('halite_2', dict(bot = 2.31, clay = None)), \
#     ('arg_halite_8', dict(bot = 4.27, clay = 'I'))])

#(These definitions come from:
#Munson, D. E. , 1997. "Constitutive Model of Creep in Rock Salt Applied to
#Underground Room Closure,"   International Journal fo Rock Mechanics, Min. Sci.
#Volume 34, Number 2, pp. 233-247. Elsevier Science Ltd.
#The same elevations appear in SAND88-2948. )
layers = OrderedDict( [\
    ('polyhalite_1', dict(bot = -54.19, clay = None)), \
    ('arg_halite_1', dict(bot = -49.99, clay = None)), \
    ('anhydrite_1', dict(bot = -30.60, clay = 'A')), \
    ('arg_halite_2', dict(bot = -26.21, clay = None)), \
    ('anhydrite_2', dict(bot = -16.41, clay = 'B')), \
    ('arg_halite_3', dict(bot = -16.33, clay = None)), \
    ('arg_halite_4', dict(bot = -11.37, clay = 'D')), \
    ('anhydrite_3', dict(bot = -8.63, clay = 'E')), \
    ('arg_halite_5', dict(bot = -7.77, clay = None)), \
    ('arg_halite_6', dict(bot = -3.72, clay = None)), \
    ('arg_halite_7', dict(bot = -2.90, clay = 'F')), \
    ('halite_1', dict(bot = 0.0, clay = 'G')), \
    ('anhydrite_4', dict(bot = 2.10, clay = 'H')), \
    ('halite_2', dict(bot = 2.31, clay = None)), \
    ('arg_halite_8', dict(bot = 4.27, clay = 'I')), \
    ('arg_halite_9', dict(bot = 6.71, clay = 'J')), \
    ('arg_halite_10', dict(bot = 7.77, clay = None)), \
    ('anhydrite_5', dict(bot = 9.16, clay = 'K')), \
    ('arg_halite_11', dict(bot = 9.35, clay = None)), \
    ('arg_halite_12', dict(bot = 10.67, clay = None)), \
    ('arg_halite_13', dict(bot = 13.58, clay = 'L')), \
    ('anhydrite_6', dict(bot = 28.3, clay = 'M')), \
    ('arg_halite_14', dict(bot = 31.86, clay = None)), \
    ('anhydrite_7', dict(bot = 49.38, clay = None))])

if seam_slip_defn == 'SAND89-2948':
    #In SAND2012-7525 and SAND89-2948 the clay seams are labeled A through M, yet they only
    #make D through L (the nine nearest the room) active.  Thus, we turn off the clay seams
    #at -30.60 m, -16.41 m, 28.30 m.
    no_clay_elevs = [-30.6, -16.41, 28.3]
    for no_clay_elev in no_clay_elevs:
        for key in layers.keys():
            if no_clay_elev == layers[key]['bot']:
                layers[key]['clay'] = None

#Switch all clay seams to non-slipping interfaces, if desired.
if not clay_seam_slip:
    for key in layers.keys():
        layers[key]['clay'] = None

#Go through and populate the top elevation keys
layers[layers.keys()[-1]]['top'] = g['top_elev']
prev_key = layers.keys()[-1]
for key in reversed(layers.keys()[:-1]):
    layers[key]['top'] = layers[prev_key]['bot']
    prev_key = key

#Generate the block
cubit.cmd('rotate -90 about X')
cubit.cmd('brick x %r y %r z %r' %(g['width']/2.0, g['thickness'], g['top_elev'] - g['bot_elev']))
cubit.cmd('move volume 1 x %r y %r z %r' \
    %(g['width']/4.0, g['thickness'] / 2.0, (g['top_elev'] - g['bot_elev'])/2.0 + g['bot_elev']))

#Cut out the room
cubit.cmd('create curve location 0 0 %r location %r 0 %r' \
    %(g['room_top_elev'], g['room_width']/2.0 - g['room_fillet_radius'], g['room_top_elev']))
cubit.cmd('create curve location %r 0 %r location %r 0 %r' \
    %(g['room_width']/2.0, g['room_top_elev'] - g['room_fillet_radius'], g['room_width']/2.0, \
    g['room_bot_elev'] + g['room_fillet_radius']))
```

```python
cubit.cmd('create curve location %r 0 %r location 0 0 %r' \
    %(g['room_width']/2.0 - g['room_fillet_radius'], g['room_bot_elev'], g['room_bot_elev']))
cubit.cmd('create curve location 0 0 %r location 0 0 %r' %(g['room_bot_elev'], g['room_top_elev']))

cubit.cmd('create surface curve 13 14 15 16')

cubit.cmd('sweep surface 7 direction 0 1 0 distance %r' %(g['thickness']))
cubit.cmd('chop volume 1 with volume 2')
cubit.cmd('delete volume 3')

#Slice the block at the layers
for key in layers.keys():
    if layers[key]['bot'] > g['bot_elev'] and layers[key]['bot'] < g['top_elev']:
        if layers[key]['clay']:
            cubit.cmd('webcut volume all with plane zplane offset %r noimprint nomerge' \
                %(layers[key]['bot']))
        else:
            cubit.cmd('webcut volume all with plane zplane offset %r imprint merge' \
                %(layers[key]['bot']))

#Add a slice at the mid height of the room, so that it is easy to extract variables along
#a line.
cubit.cmd('webcut volume all with plane zplane offset %r imprint merge' \
    %((g['room_top_elev'] + g['room_bot_elev'])/2.0))
cubit.cmd('webcut volume all with plane xplane offset 0.0 imprint merge')

#Cut volumes around room to assist with meshing
cubit.cmd('create surface ellipse major radius %r minor radius %r yplane' %(R_fine[0], R_fine[1]))
surface_list = cubit.parse_cubit_list('surface', 'with x_coord == 0 and y_coord == 0 and z_coord == 0')
ellipse_ID = surface_list[0]
cubit.cmd('move Surface ' + str(ellipse_ID) + ' location %r 0 %r include_merged' %(g['room_ctr'][0],
↪  g['room_ctr'][1]))
cubit.cmd('webcut volume all sweep surface ' + str(ellipse_ID) + ' perpendicular distance %r outward'
↪  %(g['thickness']))
cubit.cmd('delete surface ' + str(ellipse_ID))

#Imprint and merge the layers (we will add the clay seams later)
cubit.cmd('imprint all')
cubit.cmd('merge all')

#Specify the mesh

#Define some functions to help with meshing.
tol = 0.0001
def _curve_vertex_coordinates(curve_ID, ctr):
    #Order vertices based on their distance from the room
    vertex_IDs = cubit.get_relatives('curve', curve_ID, 'vertex')
    vertex_dist = []
    for vertex_ID in vertex_IDs:
        vertex = cubit.vertex(vertex_ID)
        vertex_dist.append(math.sqrt((vertex.coordinates()[0]-ctr[0])**2.0 \
            + (vertex.coordinates()[2]-ctr[1])**2.0))
    vertex_IDs = [vertex_IDs[vertex_dist.index(min(vertex_dist))], \
        vertex_IDs[vertex_dist.index(max(vertex_dist))]]
    #Get the vertex coordinates
    vertex_pos = []
    for vertex_ID in vertex_IDs:
        vertex = cubit.vertex(vertex_ID)
        vertex_pos.append(vertex.coordinates())
    return(vertex_IDs, vertex_pos)

def set_size_based_on_dist(curve_IDs, H_0, ctr, coarsen_rate, R_fine):
    for curve_ID in curve_IDs:
        [vertex_IDs, vertex_pos] = _curve_vertex_coordinates(curve_ID, ctr)
        [H_1, H_2] = _calc_size_based_on_dist(H_0, ctr, coarsen_rate, vertex_pos, R_fine)
        cubit.cmd('curve ' + str(curve_ID) + ' scheme bias fine size ' + str(H_1) \
            + ' coarse size ' + str(H_2) + ' start vertex ' + str(vertex_IDs[0]))
        cubit.cmd('curve ' + str(curve_ID) + ' interval soft')
```

```
def _calc_dists(ctr, R_fine, pt):
    #Find distance from point to the center
    ctr_2_pt_dist = math.sqrt((pt[0] - ctr[0])**2.0 + (pt[1] - ctr[1])**2.0)
    #Find angle of vertex to center of the fine region
    psi = math.atan2(pt[1] - ctr[0], pt[0] - ctr[1])
    #Find closest point on edge of fine region
    fine_pt = [ctr[0] + R_fine[0] * math.cos(psi), ctr[1] + R_fine[1] * math.sin(psi)]
    #Find the distance from center to fine point
    ctr_2_fine_pt_dist = math.sqrt((fine_pt[0] - ctr[0])**2.0 + (fine_pt[1] - ctr[1])**2.0)
    return(ctr_2_pt_dist, ctr_2_fine_pt_dist)

def  _calc_size_based_on_dist(H_0, ctr, coarsen_rate, vertex_pos, R_fine):
    #Set to H_0 inside R_fine, otherwise, calculate the element sizes based on
    #radial distance from the fine region
    [ctr_2_pt_dist, ctr_2_fine_pt_dist] = _calc_dists(ctr, R_fine, [vertex_pos[0][0], vertex_pos[0][2]])
    if ctr_2_pt_dist < ctr_2_fine_pt_dist:
        H_1 = H_0
    else:
        H_1 = H_0 * (1.0 + (ctr_2_pt_dist-ctr_2_fine_pt_dist) * coarsen_rate)
    [ctr_2_pt_dist, ctr_2_fine_pt_dist]  = _calc_dists(ctr, R_fine, [vertex_pos[1][0], vertex_pos[1][2]])
    if ctr_2_pt_dist < ctr_2_fine_pt_dist:
        H_2 = H_0
    else:
        H_2 = H_0 * (1.0 + (ctr_2_pt_dist-ctr_2_fine_pt_dist) * coarsen_rate)
    return(H_1, H_2)


#Mesh the front surfaces
curves = cubit.parse_cubit_list('curve', 'with y_coord <= ' + str(tol))
set_size_based_on_dist(curves, H_xz, g['room_ctr'], coarsen_rate, R_fine)
front_surf = cubit.parse_cubit_list('surface', 'with y_coord <= ' + str(tol))
front_surf_str = str(front_surf)[1:-1]
cubit.cmd('surface ' + front_surf_str + ' scheme pave')
#Mesh the corners of the rooms first, since they can get ugly elements sometimes
curves = cubit.parse_cubit_list('curve', 'with y_coord <= ' + str(tol) \
    + ' and with x_coord > ' + str(g['room_width'] / 2.0 - g['room_fillet_radius'] + tol) \
    + ' and with x_coord < ' + str(g['room_width'] / 2.0 - tol) \
    + ' and with z_coord > ' + str(g['room_top_elev'] - g['room_fillet_radius'] + tol) \
    + ' and with z_coord < ' + str(g['room_top_elev'] - tol))
top_corner_str = str(curves)[1:-1]
print top_corner_str
cubit.cmd('mesh curve ' + top corner_str)
cubit.cmd('mesh surface ' + front_surf_str)

#Perform a uniform refinement if requested
if split_elements:
    front_surf = cubit.parse_cubit_list('surface', 'with y_coord <= 0')
    cubit.cmd('volume all scale x 1 y 0.5 z 1')
    g['thickness'] = g['thickness'] * 0.5
    cubit.cmd('refine surface ' + str(front_surf)[1:-1] + ' numsplit 1')

#Sweep the mesh through the thickness
cubit.cmd('volume all scheme sweep vector 0 1 0')
#Set 'redistribute nodes on' so to preserve the bias during the sweep. (This does the
#same thing as the 'propagate bias' parameter in the sweep-source-target command)
cubit.cmd('volume all redistribute nodes on')

#Mesh thru the thickness
cubit.cmd('volume all size %r' %(H_y))
cubit.cmd('mesh volume all')

#Mirror and merge
cubit.cmd('volume all copy reflect x')
cubit.cmd('merge all')

#Group each layer's volumes together
for key in layers.keys():
    layers[key]['volumes'] = cubit.parse_cubit_list('volume', \
        "with z_coord >= " + str(layers[key]['bot']) \
        + " and with z_coord <= " + str(layers[key]['top']))

#Find which volumes pertain to which materials
keys = layers.keys()
material_names = []
vol_IDs = []
material_dict = dict()
for key in keys:
    match = re.match(r'(\w+)(_\d+)', key)
    material_names.append(match.group(1))
    vol_IDs.append(layers[key]['volumes'])
#Initialize the dictionary with just the unique materials
unique_material_names = set(material_names)
for unique_material_name in unique_material_names:
    material_dict[unique_material_name] = []
#Populate the dictionary
for material_name, vol_list in zip(material_names, vol_IDs):
    material_dict[material_name].extend(vol_list)

#Identify interior room surfaces
roof_surf = cubit.parse_cubit_list('surface', \
    'with z_coord <= ' + str(g['room_top_elev'] + tol) \
    + ' and with z_coord >= ' + str(g['room_top_elev']-tol) \
    + ' and with y_coord > 0.0 ' \
    + ' and with y_coord < ' + str(g['thickness']) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)))
floor_surf = cubit.parse_cubit_list('surface', \
    'with z_coord <= ' + str(g['room_bot_elev'] + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev']-tol) \
    + ' and with y_coord > 0.0 ' \
    + ' and with y_coord < ' + str(g['thickness']) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)))
right_wall_surf = cubit.parse_cubit_list('surface', \
    'with z_coord <= ' + str(g['room_top_elev'] + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
    + ' and with y_coord > 0.0 ' \
    + ' and with y_coord < ' + str(g['thickness']) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with x_coord >= ' + str(g['room_width']/2.0 - g['room_fillet_radius']))
if not x_sym:
    left_wall_surf = cubit.parse_cubit_list('surface', \
        'with z_coord <= ' + str(g['room_top_elev'] + tol) \
        + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
        + ' and with y_coord > 0.0 ' \
        + ' and with y_coord < ' + str(g['thickness']) \
        + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)) \
        + ' and with x_coord <= ' + str(-(g['room_width']/2.0 - g['room_fillet_radius'])))
room_surf = []
room_surf.extend(roof_surf)
room_surf.extend(floor_surf)
if not x_sym:
    room_surf.extend(left_wall_surf)
room_surf.extend(right_wall_surf)
room_surf_str = str(room_surf)[1:-1]

#Only add in the slip lines after the rest of the mesh has been generated
#This way, the mesh will be identical, regardless of whether you have the slip lines
#on or off.
i = 1000
j = 0
for key in layers.keys()[1:]:
    if layers[key]['clay'] is not None:
        merged_surfaces = cubit.parse_cubit_list('surface', \
            'with z_coord >= ' + str(layers[key]['bot']-tol) \
            + ' and with z_coord <= ' + str(layers[key]['bot']+tol))
        cubit.cmd('unmerge surface ' + str(merged_surfaces)[1:-1])
        #(Note: you cannot assume that the previously merged surfaces all
        #remain on the top or the bottom of the clay seam.  Cubit can randomly
        #chose which surfaces remain on the top and which are on the bottom.
        #Instead we explicitly specify that the master surfaces must be in the
        #volumes above the clay seam.)
        master_surfaces = cubit.parse_cubit_list('surface', \
            'in volume ' + str(layers[key]['volumes'])[1:-1] \
            + ' with z_coord >= ' + str(layers[key]['bot']-tol) \
            + ' and with z_coord <= ' + str(layers[key]['bot']+tol))
        #Sierra gets bent out of shape if you define a contact surface that uses
        #a surface that is already a part of another contact surface.  This can happen
        #if a clay seam happens to align with a surface in the room, and you want
        #to allow the room surfaces to have self contact.  We avoid this by making
        #sure the clay seam surfaces do not include the room surfaces.
        master_surfaces = list(set(master_surfaces) - set(room_surf))
        #Get the slave surfaces by subtracting the master surfaces and the room
        #surfaces from all the surfaces at the clay seam
        surfaces = cubit.parse_cubit_list('surface', \
            'with z_coord >= ' + str(layers[key]['bot']-tol) \
            + ' and with z_coord <= ' + str(layers[key]['bot']+tol))
        slave_surfaces = list(set(surfaces) - set(master_surfaces) - set(room_surf))
        j = j + 1
        i = i + 1
        cubit.cmd('sideset ' + str(i) + ' surface ' + str(master_surfaces)[1:-1])
        cubit.cmd('sideset ' + str(i) + ' name ' \
            + '\'SS_clay_' +  layers[key]['clay'] + '_master\'')
        i = i + 1
        cubit.cmd('sideset ' + str(i) + ' surface ' + str(slave_surfaces)[1:-1])
        cubit.cmd('sideset ' + str(i) + ' name ' \
            + '\'SS_clay_' +  layers[key]['clay'] + '_slave\'')
        #Print the results for troubleshooting purposes
        print("Slipping clay seam " + layers[key]['clay']+ " created at elevation = " + str(layers[key]['bot'])
        ↪  + " m.")
        print("Master surfaces = " + str(master_surfaces)[1:-1])
        print("Slave surfaces = " + str(slave_surfaces)[1:-1])

i = 1
if room_surf_blocks:
    for key in material_dict.keys():
        i = i + 1
        cubit.cmd('Block ' + str(i) + 'hex in volume ' + str(material_dict[key])[1:-1] + ' except hex in edge
        ↪  in surface ' + room_surf_str)
        cubit.cmd('Block ' + str(i) + ' name \'B_' + key + '\'')
    for key in material_dict.keys():
        i = i + 1
        cubit.cmd('Block ' + str(i) + 'hex in volume ' + str(material_dict[key])[1:-1] + ' in edge in surface '
        ↪  + room_surf_str)
        cubit.cmd('Block ' + str(i) + ' name \'B_' + key + '_room_surf\'')
else:
    for key in material_dict.keys():
        i = i + 1
        cubit.cmd('Block ' + str(i) + ' volume ' + str(material_dict[key])[1:-1])
        cubit.cmd('Block ' + str(i) + ' name \'B_' + key + '\'')

#Create the nodesets and sidesets
#Plane strain surfaces
front_surf = cubit.parse_cubit_list('surface', 'with y_coord <= 0')
cubit.cmd('nodeset 1 surface ' + str(front_surf)[1:-1])
cubit.cmd('nodeset 1 name \'NS_front_surf\'')
back_surf = cubit.parse_cubit_list('surface', 'with y_coord >= ' + str(g['thickness']-tol))
cubit.cmd('nodeset 2 surface ' + str(back_surf)[1:-1])
cubit.cmd('nodeset 2 name \'NS_back_surf\'')
#Left and right surfaces
if not x_sym:
    left_surf_x = g['width']/2.0
else:
    left_surf_x = 0.0
left_surf = cubit.parse_cubit_list('surface', "with x_coord <= " + str(-(g['width']/2.0 - tol)))
cubit.cmd('nodeset 3 surface ' + str(left_surf)[1:-1])
cubit.cmd('nodeset 3 name \'NS_left_surf\'')
cubit.cmd('sideset 3 surface ' + str(left_surf)[1:-1])
cubit.cmd('sideset 3 name \'SS_left_surf\'')
right_surf = cubit.parse_cubit_list('surface', 'with x_coord >= ' + str(g['width']/2.0 - tol))
cubit.cmd('nodeset 4 surface ' + str(right_surf)[1:-1])
cubit.cmd('nodeset 4 name \'NS_right_surf\'')
cubit.cmd('sideset 4 surface ' + str(right_surf)[1:-1])
cubit.cmd('sideset 4 name \'SS_right_surf\'')
#Top and bottom surfaces
bot_surf = cubit.parse_cubit_list('surface', 'with z_coord <= ' + str(g['bot_elev'] + tol))
cubit.cmd('nodeset 5 surface ' + str(bot_surf)[1:-1])
cubit.cmd('nodeset 5 name \'NS_bot_surf\'')
cubit.cmd('sideset 5 surface ' + str(bot_surf)[1:-1])
cubit.cmd('sideset 5 name \'SS_bot_surf\'')
top_surf = cubit.parse_cubit_list('surface', 'with z_coord >= ' + str(g['top_elev'] - tol))
cubit.cmd('nodeset 6 surface ' + str(top_surf)[1:-1])
cubit.cmd('nodeset 6 name \'NS_top_surf\'')
cubit.cmd('sideset 6 surface ' + str(top_surf)[1:-1])
cubit.cmd('sideset 6 name \'SS_top_surf\'')
#Interior room surface
cubit.cmd('sideset 7 surface ' + room_surf_str)
cubit.cmd('sideset 7 name \'SS_room_surf\'')
#Room nodesets for history traces
roof_vertex = cubit.parse_cubit_list('vertex', \
    'with y_coord <= 0 \
    and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_top_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_top_elev'] + tol))
cubit.cmd('nodeset 8 vertex ' + str(roof_vertex)[1:-1])
cubit.cmd('nodeset 8 name \'NS_roof_ctr\'')
floor_vertex = cubit.parse_cubit_list('vertex', \
    'with y_coord <= 0 \
    and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_bot_elev'] + tol))
cubit.cmd('nodeset 9 vertex ' + str(floor_vertex)[1:-1])
cubit.cmd('nodeset 9 name \'NS_floor_ctr\'')
right_wall_vertex = cubit.parse_cubit_list('vertex', \
    'with y_coord <= 0 \
    and with x_coord >= ' + str(g['room_width']/2.0 - tol) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol) \
    + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
cubit.cmd('nodeset 10 vertex ' + str(right_wall_vertex)[1:-1])
cubit.cmd('nodeset 10 name \'NS_right_wall_ctr\'')
if not x_sym:
    left_wall_vertex = cubit.parse_cubit_list('vertex', \
        'with y_coord <= 0 \
        and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)) \
        + ' and with x_coord <= ' + str(-(g['room_width']/2.0 - tol)) \
        + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol) \
        + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
    cubit.cmd('nodeset 11 vertex ' + str(left_wall_vertex)[1:-1])
    cubit.cmd('nodeset 11 name \'NS_left_wall_ctr\'')

roof_ctr_nodes = cubit.parse_cubit_list('node', \
    'with y_coord >= 0.0 \
    and with y_coord <= ' + str(g['thickness']) \
    + ' and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_top_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_top_elev'] + tol))
cubit.cmd('nodeset 12 node ' + str(roof_ctr_nodes)[1:-1])
cubit.cmd('nodeset 12 name \'NS_roof_ctr_new\'')

if not x_sym:
    left_wall_ctr_nodes = cubit.parse_cubit_list('node', \
        'with y_coord >= 0.0 \
        and with y_coord <= ' + str(g['thickness']) \
        + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)) \
        + ' and with x_coord <= ' + str(-(g['room_width']/2.0 - tol)) \
        + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol) \
        + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
    cubit.cmd('nodeset 13 node ' + str(left_wall_ctr_nodes)[1:-1])
    cubit.cmd('nodeset 13 name \'NS_left_wall_ctr_new\'')

right_wall_ctr_nodes = cubit.parse_cubit_list('node', \
    'with y_coord >= 0.0 \
    and with y_coord <= ' + str(g['thickness']) \
    + ' and with x_coord >= ' + str((g['room_width']/2.0 - tol)) \
    + ' and with x_coord <= ' + str((g['room_width']/2.0 + tol)) \
    + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol)\
    + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
cubit.cmd('nodeset 14 node ' + str(right_wall_ctr_nodes)[1:-1])
cubit.cmd('nodeset 14 name \'NS_right_wall_ctr_new\'')


floor_ctr_nodes = cubit.parse_cubit_list('node', \
    'with y_coord >= 0.0 \
    and with y_coord <= ' + str(g['thickness']) \
    + ' and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_bot_elev'] + tol))
cubit.cmd('nodeset 15 node ' + str(floor_ctr_nodes)[1:-1])
cubit.cmd('nodeset 15 name \'NS_floor_ctr_new\'')

#Move mesh so it is centered on y=0.
cubit.cmd('move volume all x 0.0 y %r z 0.0 include_merged' %(-g['thickness'] / 2.0))

#Export the Mesh
import os
pwd = os.getcwd()
export_path = os.path.join(pwd, mesh_name + '-%d_pop_row_deep-%d_elements_2017_11_28.g' %(num_CCO_rows,
↪  g['room_width'] / H_xz))
print export_path
cubit.cmd('export genesis \'%s\' overwrite' % export_path)
if room_surf_blocks:
    export_path = os.path.join(pwd, mesh_name + '-room_surf-%d_pop_row_deep-%d_elements_2017_11_28.g'
    ↪  %(num_CCO_rows, g['room_width'] / H_xz))
    cubit.cmd("export genesis \'%s\' block 6 7 overwrite" % export_path)

#Save the geometry values
import cPickle
#cPickle can save as a binary or a text file.  Binaries are smaller, so I
#use them here.  The 'wb' option says to open the binary file with write access.
geom_path = os.path.join(pwd, mesh_name + '-%d_pop_row_deep-geom_values_2017_11_28.pkl' %(num_CCO_rows))
geom_file = open(geom_path,'wb')
#Save the file.  Protocol 2 is the faster binary protocol for Python 2.7
cPickle.dump(g, geom_file, protocol = 2)
geom_file.close()

#Print the pressures at the top and bottom surfaces
p1 = -15.97e6
p2 = -13.57e6
z1 = -54.19
z2 = 52.87
p = lambda z: (p2 - p1) / (z2 - z1) * (z - z1) + p1
print 'At z = ' + str(g['bot_elev']) + ' p = ' + str(p(g['bot_elev']))
print 'At z = ' + str(g['top_elev']) + ' p = ' + str(p(g['top_elev']))
```

```
#Print the error count
error_new = cubit.get_error_count()
print 'Cubit error count = %r' %(error_new - error_old)
```

Roof Fall Compaction, Disposal Room

```
#!python
#Cubit v15.0, and v15.3 (v15.1 and v15.2 don't always mesh the surfaces)
cubit.cmd('reset')
cubit.set_playback_paused_on_error(False)
error_old = cubit.get_error_count()
cubit.cmd('undo off')
import math
from collections import OrderedDict
import re

############################################
# create WIPP lower Horizon geologic mesh
# with roof fall geometry
############################################

# Define conversion factors
in_m = 0.0254

mesh_name = 'disposal_room-with_clay_slip-lower_horizon-rockFall'

#Initialize dictionary to store geometry values
#(All dimensions are in meters.)
g = dict()

#Define whether to model clay seams as slipping surfaces
#clay_seam_slip = False
clay_seam_slip = True
#Define whether to split each element into 4 elements
split_elements = False
#Define the clay seam slip definition
#seam_slip_defn = 'SAND89-2948'
seam_slip_defn = 'all'
#Define whether to create blocks for the layer of elements on the interior of room
room_surf_blocks = False
#Define whether to assume symmetry across the X=0 plane
#x_sym = True
x_sym = False

#Define the room dimensions as defined in SAND97-0795
#(elevations relative to clay seam G)
#(Figure 3 in SAND97-0795 has the room top elevation as -1.96 m and the bottom as -5.92 m,
#while Figure 4 has the room top elevation at -2.43 m and the bottom at -6.39 m.  This discrepancy
#is discussed in a March 4 memo from Charles Stone to B. Butcher, which is in Appendix B-8 of
#SAND97-0796.  Apparently they decided to move the room elevation because they felt it was most
#important to have the distance between marker band 139 and the disposal room floor be 1.39 m.)
g['room_top_elev'] = -2.43
g['room_bot_elev'] = -6.39
g['room_width'] = 5.03 * 2.0
#(This fillet radius is from the cutting head radius used in room D, plus a small amount.  I do not
#know if the disposal rooms used the same cutting head as Room D, but it seems like a safe bet.)
#g['room_fillet_radius'] = 0.00001 # 18.0 * 25.4 / 1000.0

#JEB- using Cubit 15-4 with the original room_fillet_radius causes some elements at the corner of the room
#to have nearly coincident nodes which results in Sierra/SM computing a critical time step O(10^-9) sec.
#Setting the radius to zero eliminates the problem.
g['room_fillet_radius'] =0.0

#Define the coordinates of the center of the room
g['room_ctr'] = [0.0, (g['room_bot_elev'] + g['room_top_elev']) / 2.0]
```

```
#Define the total simulation dimensions as defined in SAND97-0795
#(elevations relative to the bottom of the bottom halite layer)
g['top_elev'] = 52.87
g['bot_elev'] = -54.19
g['width'] = 20.27 * 2.0

CCO_gap = 5.0 / 1000.0
d_t = 1.50 / 1000
CCO_dia = (22.50 * in_m) + 2.0*d_t # JLB Detailed CCO
seven_pack_depth = CCO_dia + (CCO_dia + CCO_gap) * math.cos(math.pi/6) * 2.0 + CCO_gap  # JLB - Added CCO_gap
num_CCO_rows = 1.0
CCO_row_depth = seven_pack_depth + CCO_dia # add extra drum diameter
g['thickness'] = CCO_row_depth * num_CCO_rows

print CCO_dia
print CCO_row_depth

#Define the fine element size in the X-Z plane
H_xz = g['room_width'] / 24.0
#Define the fine element size in the Y-direction
H_y = H_xz * 1.0
print H_y
#Define the radius of the fine region
R_fine = [g['room_width'] * 1.25, (g['room_top_elev'] - g['room_bot_elev']) * 2]
#Define the coarsening rate
coarsen_rate = 0.36

#Define stratigraphy
#
#(bot is the bottom of each layer, relative to the bottom of the halite in the room.
#Clay defines whether there is a sliding layer of clay at the bottom.)
#
#(These definitions come from SAND97-0795)
#layers = OrderedDict( [\
#    ('arg_halite_1', dict(bot = -54.19, clay = None)), \
#    ('anhydrite_1', dict(bot = -8.63, clay = 'E')), \
#    ('arg_halite_2', dict(bot = -7.77, clay = None)), \
#    ('halite_1', dict(bot = 0.0, clay = 'G')), \
#    ('anhydrite_2', dict(bot = 2.10, clay = 'H')), \
#    ('halite_2', dict(bot = 2.31, clay = None)), \
#    ('arg_halite_8', dict(bot = 4.27, clay = 'I'))])

#(These definitions come from:
#Munson, D. E. , 1997. "Constitutive Model of Creep in Rock Salt Applied to
#Underground Room Closure,"   International Journal fo Rock Mechanics, Min. Sci.
#Volume 34, Number 2, pp. 233-247. Elsevier Science Ltd.
#The same elevations appear in SAND88-2948. )
layers = OrderedDict( [\
    ('polyhalite_1', dict(bot = -54.19, clay = None)), \
    ('arg_halite_1', dict(bot = -49.99, clay = None)), \
    ('anhydrite_1', dict(bot = -30.60, clay = 'A')), \
    ('arg_halite_2', dict(bot = -26.21, clay = None)), \
    ('anhydrite_2', dict(bot = -16.41, clay = 'B')), \
    ('arg_halite_3', dict(bot = -16.33, clay = None)), \
    ('arg_halite_4', dict(bot = -11.37, clay = 'D')), \
    ('anhydrite_3', dict(bot = -8.63, clay = 'E')), \
    ('arg_halite_5', dict(bot = -7.77, clay = None)), \
    ('arg_halite_6', dict(bot = -3.72, clay = None)), \
    ('arg_halite_7', dict(bot = -2.90, clay = 'F')), \
    ('halite_1', dict(bot = 0.0, clay = 'G')), \
    ('anhydrite_4', dict(bot = 2.10, clay = 'H')), \
    ('halite_2', dict(bot = 2.31, clay = None)), \
    ('arg_halite_8', dict(bot = 4.27, clay = 'I')), \
    ('arg_halite_9', dict(bot = 6.71, clay = 'J')), \
    ('arg_halite_10', dict(bot = 7.77, clay = None)), \
    ('anhydrite_5', dict(bot = 9.16, clay = 'K')), \
    ('arg_halite_11', dict(bot = 9.35, clay = None)), \
    ('arg_halite_12', dict(bot = 10.67, clay = None)), \
    ('arg_halite_13', dict(bot = 13.58, clay = 'L')), \
    ('anhydrite_6', dict(bot = 28.3, clay = 'M')), \
    ('arg_halite_14', dict(bot = 31.86, clay = None)), \
    ('anhydrite_7', dict(bot = 49.38, clay = None))])

if seam_slip_defn == 'SAND89-2948':
    #In SAND2012-7525 and SAND89-2948 the clay seams are labeled A through M, yet they only
    #make D through L (the nine nearest the room) active.  Thus, we turn off the clay seams
    #at -30.60 m, -16.41 m, 28.30 m.
    no_clay_elevs = [-30.6, -16.41, 28.3]
    for no_clay_elev in no_clay_elevs:
        for key in layers.keys():
            if no_clay_elev == layers[key]['bot']:
                layers[key]['clay'] = None

#Switch all clay seams to non-slipping interfaces, if desired.
if not clay_seam_slip:
    for key in layers.keys():
        layers[key]['clay'] = None

#Go through and populate the top elevation keys
layers[layers.keys()[-1]]['top'] = g['top_elev']
prev_key = layers.keys()[-1]
for key in reversed(layers.keys()[:-1]):
    layers[key]['top'] = layers[prev_key]['bot']
    prev_key = key

#Generate the block
cubit.cmd('rotate -90 about X')
cubit.cmd('brick x %r y %r z %r' %(g['width']/2.0, g['thickness'], g['top_elev'] - g['bot_elev']))
cubit.cmd('move volume 1 x %r y %r z %r' \
    %(g['width']/4.0, g['thickness'] / 2.0, (g['top_elev'] - g['bot_elev'])/2.0 + g['bot_elev']))

#Cut out the room
cubit.cmd('create curve location 0 0 %r location %r 0 %r' \
    %(g['room_top_elev'], g['room_width']/2.0 - g['room_fillet_radius'], g['room_top_elev']))
cubit.cmd('create curve location %r 0 %r location %r 0 %r' \
    %(g['room_width']/2.0, g['room_top_elev'] - g['room_fillet_radius'], g['room_width']/2.0, \
    g['room_bot_elev'] + g['room_fillet_radius']))
cubit.cmd('create curve location %r 0 %r location 0 0 %r' \
    %(g['room_width']/2.0 - g['room_fillet_radius'], g['room_bot_elev'], g['room_bot_elev']))
cubit.cmd('create curve location 0 0 %r location 0 0 %r' %(g['room_bot_elev'], g['room_top_elev']))

cubit.cmd('create surface curve 13 14 15 16')

cubit.cmd('sweep surface 7 direction 0 1 0 distance %r' %(g['thickness']))
cubit.cmd('chop volume 1 with volume 2')
cubit.cmd('delete volume 3')

#Slice the block at the layers
for key in layers.keys():
    if layers[key]['bot'] > g['bot_elev'] and layers[key]['bot'] < g['top_elev']:
        if layers[key]['clay']:
            cubit.cmd('webcut volume all with plane zplane offset %r noimprint nomerge' \
                %(layers[key]['bot']))
        else:
            cubit.cmd('webcut volume all with plane zplane offset %r imprint merge' \
                %(layers[key]['bot']))

#Add a slice at the mid height of the room, so that it is easy to extract variables along
#a line.
cubit.cmd('webcut volume all with plane zplane offset %r imprint merge' \
    %((g['room_top_elev'] + g['room_bot_elev'])/2.0))
cubit.cmd('webcut volume all with plane xplane offset 0.0 imprint merge')

#Cut volumes around room to assist with meshing
cubit.cmd('create surface ellipse major radius %r minor radius %r yplane' %(R_fine[0], R_fine[1]))
```

```
surface_list = cubit.parse_cubit_list('surface', 'with x_coord == 0 and y_coord == 0 and z_coord == 0')
ellipse_ID = surface_list[0]
cubit.cmd('move Surface ' + str(ellipse_ID) + ' location %r 0 %r include_merged' %(g['room_ctr'][0],
↪  g['room_ctr'][1]))
cubit.cmd('webcut volume all sweep surface ' + str(ellipse_ID) + ' perpendicular distance %r outward'
↪  %(g['thickness']))
cubit.cmd('delete surface ' + str(ellipse_ID))

# Start JEB's changes
#Add webcuts to remove the region above the room.  It will be added back in later.
cubit.cmd('webcut volume 35  with plane normal to curve 37  close_to vertex 31  ')
cubit.cmd('delete volume 40')
# End JEB's changes

#Imprint and merge the layers (we will add the clay seams later)
cubit.cmd('imprint all')
cubit.cmd('merge all')

#Specify the mesh

#Define some functions to help with meshing.
tol = 0.0001
def _curve_vertex_coordinates(curve_ID, ctr):
    #Order vertices based on their distance from the room
    vertex_IDs = cubit.get_relatives('curve', curve_ID, 'vertex')
    vertex_dist = []
    for vertex_ID in vertex_IDs:
        vertex = cubit.vertex(vertex_ID)
        vertex_dist.append(math.sqrt((vertex.coordinates()[0]-ctr[0])**2.0 \
            + (vertex.coordinates()[2]-ctr[1])**2.0))
    vertex_IDs = [vertex_IDs[vertex_dist.index(min(vertex_dist))], \
        vertex_IDs[vertex_dist.index(max(vertex_dist))]]
    #Get the vertex coordinates
    vertex_pos = []
    for vertex_ID in vertex_IDs:
        vertex = cubit.vertex(vertex_ID)
        vertex_pos.append(vertex.coordinates())
    return(vertex_IDs, vertex_pos)

def set_size_based_on_dist(curve_IDs, H_0, ctr, coarsen_rate, R_fine):
    for curve_ID in curve_IDs:
        [vertex_IDs, vertex_pos] = _curve_vertex_coordinates(curve_ID, ctr)
        [H_1, H_2] = _calc_size_based_on_dist(H_0, ctr, coarsen_rate, vertex_pos, R_fine)
        cubit.cmd('curve ' + str(curve_ID) + ' scheme bias fine size ' + str(H_1) \
            + ' coarse size ' + str(H_2) + ' start vertex ' + str(vertex_IDs[0]))
        cubit.cmd('curve ' + str(curve_ID) + ' interval soft')

def _calc_dists(ctr, R_fine, pt):
    #Find distance from point to the center
    ctr_2_pt_dist = math.sqrt((pt[0] - ctr[0])**2.0 + (pt[1] - ctr[1])**2.0)
    #Find angle of vertex to center of the fine region
    psi = math.atan2(pt[1] - ctr[0], pt[0] - ctr[1])
    #Find closest point on edge of fine region
    fine_pt = [ctr[0] + R_fine[0] * math.cos(psi), ctr[1] + R_fine[1] * math.sin(psi)]
    #Find the distance from center to fine point
    ctr_2_fine_pt_dist = math.sqrt((fine_pt[0] - ctr[0])**2.0 + (fine_pt[1] - ctr[1])**2.0)
    return(ctr_2_pt_dist, ctr_2_fine_pt_dist)

def  _calc_size_based_on_dist(H_0, ctr, coarsen_rate, vertex_pos, R_fine):
    #Set to H_0 inside R_fine, otherwise, calculate the element sizes based on
    #radial distance from the fine region
    [ctr_2_pt_dist, ctr_2_fine_pt_dist] = _calc_dists(ctr, R_fine, [vertex_pos[0][0], vertex_pos[0][2]])
    if ctr_2_pt_dist < ctr_2_fine_pt_dist:
        H_1 = H_0
    else:
        H_1 = H_0 * (1.0 + (ctr_2_pt_dist-ctr_2_fine_pt_dist) * coarsen_rate)
    [ctr_2_pt_dist, ctr_2_fine_pt_dist]  = _calc_dists(ctr, R_fine, [vertex_pos[1][0], vertex_pos[1][2]])
    if ctr_2_pt_dist < ctr_2_fine_pt_dist:
        H_2 = H_0
    else:
        H_2 = H_0 * (1.0 + (ctr_2_pt_dist-ctr_2_fine_pt_dist) * coarsen_rate)
    return(H_1, H_2)


#Mesh the front surfaces
curves = cubit.parse_cubit_list('curve', 'with y_coord <= ' + str(tol))
set_size_based_on_dist(curves, H_xz, g['room_ctr'], coarsen_rate, R_fine)
front_surf = cubit.parse_cubit_list('surface', 'with y_coord <= ' + str(tol))
front_surf_str = str(front_surf)[1:-1]
cubit.cmd('surface ' + front_surf_str + ' scheme pave')
#Mesh the corners of the rooms first, since they can get ugly elements sometimes
curves = cubit.parse_cubit_list('curve', 'with y_coord <= ' + str(tol) \
    + ' and with x_coord > ' + str(g['room_width'] / 2.0 - g['room_fillet_radius'] + tol) \
    + ' and with x_coord < ' + str(g['room_width'] / 2.0 - tol) \
    + ' and with z_coord > ' + str(g['room_top_elev'] - g['room_fillet_radius'] + tol) \
    + ' and with z_coord < ' + str(g['room_top_elev'] - tol))
top_corner_str = str(curves)[1:-1]
print top_corner_str
cubit.cmd('mesh curve ' + top corner_str)
cubit.cmd('mesh surface ' + front_surf_str)

#Perform a uniform refinement if requested
if split_elements:
    front_surf = cubit.parse_cubit_list('surface', 'with y_coord <= 0')
    cubit.cmd('volume all scale x 1 y 0.5 z 1')
    g['thickness'] = g['thickness'] * 0.5
    cubit.cmd('refine surface ' + str(front_surf)[1:-1] + ' numsplit 1')

#Sweep the mesh through the thickness
cubit.cmd('volume all scheme sweep vector 0 1 0')
#Set 'redistribute nodes on' so to preserve the bias during the sweep. (This does the
#same thing as the 'propagate bias' parameter in the sweep-source-target command)
cubit.cmd('volume all redistribute nodes on')

#Mesh thru the thickness
cubit.cmd('volume all size %r' %(H_y))
cubit.cmd('mesh volume all')

#Mirror and merge
cubit.cmd('volume all copy reflect x')
cubit.cmd('merge all')

#JEB- Create block above room, part of which will be allowed to fall into the room
#For now just hardwire some of the parameter values.  May need to fully paramererize to make
#more general in the future.
# rockfall height = distance between top of room and clay seam G
g['rockfall_height'] = 2.43
cubit.cmd('brick x %r y %r z %r' %(g['room_width'], g['thickness'],  g['rockfall_height']))
cubit.cmd('move volume 76 x 0 y %r z %r' %( 0.5*g['thickness'], -0.5*g['rockfall_height']))

#room width = 10.06 m
# fracture point measured from left room boundary = ~8 ft = 2.44m
fracture_pt = 2.44
fraction = fracture_pt / g['room_width']
cubit.cmd('create vertex on curve 876  fraction %r from start'  %(fraction))

cubit.cmd('create curve vertex 451 454')

cubit.cmd('webcut volume 76  sweep curve 885  vector 0 1 0 through_all ')

#create a vertex on the volume which will be the center of the roof- may need this to determine
#the vertical closure
cubit.cmd('create vertex on curve 878 fraction .5 from start ')
cubit.cmd('create vertex on curve 880 fraction .5 from start ')
cubit.cmd('create curve vertex 462 463')
cubit.cmd('imprint volume 76  with curve 897')
```

```
#cubit.cmd('webcut volume 76 with plane normal to curve 899 fraction 0.01 from vertex 452')
cubit.cmd('webcut volume 76 with plane normal to curve 899 fraction 0.05 from vertex 452')
cubit.cmd('webcut volume 78 with plane normal to curve 902 fraction 0.05 from vertex 451')
cubit.cmd('delete Volume 76 Volume 79' )

cubit.cmd('surface 572  size .4' )
cubit.cmd('surface 572 Scheme pave' )
cubit.cmd('mesh surface 572' )
cubit.cmd('surface 555 size .4' )
cubit.cmd('surface 555 Scheme pave' )
cubit.cmd('mesh surface 555' )

#Sweep the mesh through the thickness
cubit.cmd('volume 77 78  scheme sweep vector 0 1 0')
#Set 'redistribute nodes on' so to preserve the bias during the sweep. (This does the
#same thing as the 'propagate bias' parameter in the sweep-source-target command)
cubit.cmd('volume all redistribute nodes on')

#Mesh thru the thickness
cubit.cmd('volume 77 78  size %r' %(H_y))
cubit.cmd('mesh volume 77 78')

#Group each layer's volumes together
for key in layers.keys():
    layers[key]['volumes'] = cubit.parse_cubit_list('volume', \
        "with z_coord >= " + str(layers[key]['bot']) \
        + " and with z_coord <= " + str(layers[key]['top']))

#Find which volumes pertain to which materials
keys = layers.keys()
material_names = []
vol_IDs = []
material_dict = dict()
for key in keys:
    match = re.match(r'(\w+)(_\d+)', key)
    material_names.append(match.group(1))
    vol_IDs.append(layers[key]['volumes'])
#Initialize the dictionary with just the unique materials
unique_material_names = set(material_names)
for unique_material_name in unique_material_names:
    material_dict[unique_material_name] = []
#Populate the dictionary
for material_name, vol_list in zip(material_names, vol_IDs):
    material_dict[material_name].extend(vol_list)

#Identify interior room surfaces
roof_surf = cubit.parse_cubit_list('surface', \
    'with z_coord <= ' + str(g['room_top_elev'] + tol) \
    + ' and with z_coord >= ' + str(g['room_top_elev']-tol) \
    + ' and with y_coord > 0.0 ' \
    + ' and with y_coord < ' + str(g['thickness']) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)))
floor_surf = cubit.parse_cubit_list('surface', \
    'with z_coord <= ' + str(g['room_bot_elev'] + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev']-tol) \
    + ' and with y_coord > 0.0 ' \
    + ' and with y_coord < ' + str(g['thickness']) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)))
right_wall_surf = cubit.parse_cubit_list('surface', \
    'with z_coord <= ' + str(g['room_top_elev'] + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
    + ' and with y_coord > 0.0 ' \
    + ' and with y_coord < ' + str(g['thickness']) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with x_coord >= ' + str(g['room_width']/2.0 - g['room_fillet_radius']))
```

```
if not x_sym:
    left_wall_surf = cubit.parse_cubit_list('surface', \
        'with z_coord <= ' + str(g['room_top_elev'] + tol) \
        + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
        + ' and with y_coord > 0.0 ' \
        + ' and with y_coord < ' + str(g['thickness']) \
        + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)) \
        + ' and with x_coord <= ' + str(-(g['room_width']/2.0 - g['room_fillet_radius'])))
room_surf = []
room_surf.extend(roof_surf)
room_surf.extend(floor_surf)
if not x_sym:
    room_surf.extend(left_wall_surf)
room_surf.extend(right_wall_surf)
room_surf_str = str(room_surf)[1:-1]

#Only add in the slip lines after the rest of the mesh has been generated
#This way, the mesh will be identical, regardless of whether you have the slip lines
#on or off.
i = 1000
j = 0
for key in layers.keys()[1:]:
    if layers[key]['clay'] is not None:
        merged_surfaces = cubit.parse_cubit_list('surface', \
            'with z_coord >= ' + str(layers[key]['bot']-tol) \
            + ' and with z_coord <= ' + str(layers[key]['bot']+tol))
        cubit.cmd('unmerge surface ' + str(merged_surfaces)[1:-1])
        #(Note: you cannot assume that the previously merged surfaces all
        #remain on the top or the bottom of the clay seam.  Cubit can randomly
        #chose which surfaces remain on the top and which are on the bottom.
        #Instead we explicitly specify that the master surfaces must be in the
        #volumes above the clay seam.)
        master_surfaces = cubit.parse_cubit_list('surface', \
            'in volume ' + str(layers[key]['volumes'])[1:-1] \
            + ' with z_coord >= ' + str(layers[key]['bot']-tol) \
            + ' and with z_coord <= ' + str(layers[key]['bot']+tol))
        #Sierra gets bent out of shape if you define a contact surface that uses
        #a surface that is already a part of another contact surface.  This can happen
        #if a clay seam happens to align with a surface in the room, and you want
        #to allow the room surfaces to have self contact.  We avoid this by making
        #sure the clay seam surfaces do not include the room surfaces.
        master_surfaces = list(set(master_surfaces) - set(room_surf))
        #Get the slave surfaces by subtracting the master surfaces and the room
        #surfaces from all the surfaces at the clay seam
        surfaces = cubit.parse_cubit_list('surface', \
            'with z_coord >= ' + str(layers[key]['bot']-tol) \
            + ' and with z_coord <= ' + str(layers[key]['bot']+tol))
        slave_surfaces = list(set(surfaces) - set(master_surfaces) - set(room_surf))
        j = j + 1
        i = i + 1
        cubit.cmd('sideset ' + str(i) + ' surface ' + str(master_surfaces)[1:-1])
        cubit.cmd('sideset ' + str(i) + ' name ' \
            + '\'SS_clay_' +  layers[key]['clay'] + '_master\'')
        i = i + 1
        cubit.cmd('sideset ' + str(i) + ' surface ' + str(slave_surfaces)[1:-1])
        cubit.cmd('sideset ' + str(i) + ' name ' \
            + '\'SS_clay_' +  layers[key]['clay'] + '_slave\'')
        #Print the results for troubleshooting purposes
        print("Slipping clay seam " + layers[key]['clay']+ " created at elevation = " + str(layers[key]['bot'])
        ↪  + " m.")
        print("Master surfaces = " + str(master_surfaces)[1:-1])
        print("Slave surfaces = " + str(slave_surfaces)[1:-1])

i = 1
if room_surf_blocks:
    for key in material_dict.keys():
        i = i + 1
        cubit.cmd('Block ' + str(i) + 'hex in volume ' + str(material_dict[key])[1:-1] + ' except hex in edge
        ↪  in surface ' + room_surf_str)
        cubit.cmd('Block ' + str(i) + ' name \'B_' + key + '\'')
    for key in material_dict.keys():
        i = i + 1
        cubit.cmd('Block ' + str(i) + 'hex in volume ' + str(material_dict[key])[1:-1] + ' in edge in surface '
        ↪  + room_surf_str)
        cubit.cmd('Block ' + str(i) + ' name \'B_' + key + '_room_surf\'')
else:
    for key in material_dict.keys():
        i = i + 1
        cubit.cmd('Block ' + str(i) + ' volume ' + str(material_dict[key])[1:-1])
        cubit.cmd('Block ' + str(i) + ' name \'B_' + key + '\'')


#JEB I think we need to remove the rockfall volumes from the B_arg_halite block
cubit.cmd('block 3  remove volume 77 78 ')

cubit.cmd('set duplicate block elements off')

cubit.cmd('block 6 add volume 77')
cubit.cmd('block 6 name \'Rockfall_upper_tri\'')
cubit.cmd('block 7 volume 78')
cubit.cmd('block 7 name \'RockFall_lower\'')

#Create the nodesets and sidesets
#Plane strain surfaces
front_surf = cubit.parse_cubit_list('surface', 'with y_coord <= 0')
cubit.cmd('nodeset 1 surface ' + str(front_surf)[1:-1])
cubit.cmd('nodeset 1 name \'NS_front_surf\'')
back_surf = cubit.parse_cubit_list('surface', 'with y_coord >= ' + str(g['thickness']-tol))
cubit.cmd('nodeset 2 surface ' + str(back_surf)[1:-1])
cubit.cmd('nodeset 2 name \'NS_back_surf\'')
#Left and right surfaces
if not x_sym:
    left_surf_x = g['width']/2.0
else:
    left_surf_x = 0.0
left_surf = cubit.parse_cubit_list('surface', "with x_coord <= " + str(-(g['width']/2.0 - tol)))
cubit.cmd('nodeset 3 surface ' + str(left_surf)[1:-1])
cubit.cmd('nodeset 3 name \'NS_left_surf\'')
cubit.cmd('sideset 3 surface ' + str(left_surf)[1:-1])
cubit.cmd('sideset 3 name \'SS_left_surf\'')
right_surf = cubit.parse_cubit_list('surface', 'with x_coord >= ' + str(g['width']/2.0 - tol))
cubit.cmd('nodeset 4 surface ' + str(right_surf)[1:-1])
cubit.cmd('nodeset 4 name \'NS_right_surf\'')
cubit.cmd('sideset 4 surface ' + str(right_surf)[1:-1])
cubit.cmd('sideset 4 name \'SS_right_surf\'')
#Top and bottom surfaces
bot_surf = cubit.parse_cubit_list('surface', 'with z_coord <= ' + str(g['bot_elev'] + tol))
cubit.cmd('nodeset 5 surface ' + str(bot_surf)[1:-1])
cubit.cmd('nodeset 5 name \'NS_bot_surf\'')
cubit.cmd('sideset 5 surface ' + str(bot_surf)[1:-1])
cubit.cmd('sideset 5 name \'SS_bot_surf\'')
top_surf = cubit.parse_cubit_list('surface', 'with z_coord >= ' + str(g['top_elev'] - tol))
cubit.cmd('nodeset 6 surface ' + str(top_surf)[1:-1])
cubit.cmd('nodeset 6 name \'NS_top_surf\'')
cubit.cmd('sideset 6 surface ' + str(top_surf)[1:-1])
cubit.cmd('sideset 6 name \'SS_top_surf\'')
#Interior room surface
cubit.cmd('sideset 7 surface ' + room_surf_str)
cubit.cmd('sideset 7 name \'SS_room_surf\'')

#JEB add in clipped edges to room surface sideset
#Maybe this will solve the contact problems I have been seeing
cubit.cmd('Sideset 7 add surface 564 569')
cubit.cmd('Sideset 7 add surface 525')
```

```
# JEB Define master and slave surfaces between arg halite and
# roof fall triangles (will be tied contact)
cubit.cmd('Sideset 1025 add surface 116')
cubit.cmd('Sideset 1025 name \'SS_Halite_M1\'')
cubit.cmd('Sideset 1026 add surface 548')
cubit.cmd('Sideset 1026 name \'SS_Halite_S1\'')

cubit.cmd('Sideset 1027 add surface 553')
cubit.cmd('Sideset 1027 name \'SS_Halite_M2\'')
cubit.cmd('Sideset 1028 add surface 573')
cubit.cmd('Sideset 1028 name \'SS_Halite_S2\'')


#Room nodesets for history traces
roof_vertex = cubit.parse_cubit_list('vertex', \
    'with y_coord <= 0 \
    and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_top_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_top_elev'] + tol))
cubit.cmd('nodeset 8 vertex ' + str(roof_vertex)[1:-1])
cubit.cmd('nodeset 8 name \'NS_roof_ctr\'')
floor_vertex = cubit.parse_cubit_list('vertex', \
    'with y_coord <= 0 \
    and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_bot_elev'] + tol))
cubit.cmd('nodeset 9 vertex ' + str(floor_vertex)[1:-1])
cubit.cmd('nodeset 9 name \'NS_floor_ctr\'')
right_wall_vertex = cubit.parse_cubit_list('vertex', \
    'with y_coord <= 0 \
    and with x_coord >= ' + str(g['room_width']/2.0 - tol) \
    + ' and with x_coord <= ' + str(g['room_width']/2.0 + tol) \
    + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol) \
    + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
cubit.cmd('nodeset 10 vertex ' + str(right_wall_vertex)[1:-1])
cubit.cmd('nodeset 10 name \'NS_right_wall_ctr\'')
if not x_sym:
    left_wall_vertex = cubit.parse_cubit_list('vertex', \
        'with y_coord <= 0 \
        and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)) \
        + ' and with x_coord <= ' + str(-(g['room_width']/2.0 - tol)) \
        + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol) \
        + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
    cubit.cmd('nodeset 11 vertex ' + str(left_wall_vertex)[1:-1])
    cubit.cmd('nodeset 11 name \'NS_left_wall_ctr\'')

roof_ctr_nodes = cubit.parse_cubit_list('node', \
    'with y_coord >= 0.0 \
    and with y_coord <= ' + str(g['thickness']) \
    + ' and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_top_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_top_elev'] + tol))
cubit.cmd('nodeset 12 node ' + str(roof_ctr_nodes)[1:-1])
cubit.cmd('nodeset 12 name \'NS_roof_ctr_new\'')

if not x_sym:
    left_wall_ctr_nodes = cubit.parse_cubit_list('node', \
        'with y_coord >= 0.0 \
        and with y_coord <= ' + str(g['thickness']) \
        + ' and with x_coord >= ' + str(-(g['room_width']/2.0 + tol)) \
        + ' and with x_coord <= ' + str(-(g['room_width']/2.0 - tol)) \
        + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol) \
        + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
    cubit.cmd('nodeset 13 node ' + str(left_wall_ctr_nodes)[1:-1])
    cubit.cmd('nodeset 13 name \'NS_left_wall_ctr_new\'')
```

```
right_wall_ctr_nodes = cubit.parse_cubit_list('node', \
    'with y_coord >= 0.0 \
    and with y_coord <= ' + str(g['thickness']) \
    + ' and with x_coord >= ' + str((g['room_width']/2.0 - tol)) \
    + ' and with x_coord <= ' + str((g['room_width']/2.0 + tol)) \
    + ' and with z_coord >= ' + str(g['room_ctr'][1] - tol)\
    + ' and with z_coord <= ' + str(g['room_ctr'][1] + tol))
cubit.cmd('nodeset 14 node ' + str(right_wall_ctr_nodes)[1:-1])
cubit.cmd('nodeset 14 name \'NS_right_wall_ctr_new\'')


floor_ctr_nodes = cubit.parse_cubit_list('node', \
    'with y_coord >= 0.0 \
    and with y_coord <= ' + str(g['thickness']) \
    + ' and with x_coord >= ' + str(0 - tol) \
    + ' and with x_coord <= ' + str(0 + tol) \
    + ' and with z_coord >= ' + str(g['room_bot_elev'] - tol) \
    + ' and with z_coord <= ' + str(g['room_bot_elev'] + tol))
cubit.cmd('nodeset 15 node ' + str(floor_ctr_nodes)[1:-1])
cubit.cmd('nodeset 15 name \'NS_floor_ctr_new\'')

#Move mesh so it is centered on y=0.
cubit.cmd('move volume all x 0.0 y %r z 0.0 include_merged' %(-g['thickness'] / 2.0))

#Export the Mesh
import os
pwd = os.getcwd()
export_path = os.path.join(pwd, mesh_name + '.g')
print export_path
cubit.cmd('export genesis \'%s\' overwrite' % export_path)
if room_surf_blocks:
    export_path = os.path.join(pwd, mesh_name)
    cubit.cmd("export genesis \'%s\' block 6 7 overwrite" % export_path)

#Save the geometry values
import cPickle
#cPickle can save as a binary or a text file.  Binaries are smaller, so I
#use them here.  The 'wb' option says to open the binary file with write access.
geom_path = os.path.join(pwd, mesh_name + '.pkl')
geom_file = open(geom_path,'wb')
#Save the file.  Protocol 2 is the faster binary protocol for Python 2.7
cPickle.dump(g, geom_file, protocol = 2)
geom_file.close()

#Print the pressures at the top and bottom surfaces
p1 = -15.97e6
p2 = -13.57e6
z1 = -54.19
z2 = 52.87
p = lambda z: (p2 - p1) / (z2 - z1) * (z - z1) + p1
print 'At z = ' + str(g['bot_elev']) + ' p = ' + str(p(g['bot_elev']))
print 'At z = ' + str(g['top_elev']) + ' p = ' + str(p(g['top_elev']))

#Print the error count
error_new = cubit.get_error_count()
print 'Cubit error count = %r' %(error_new - error_old)
```

Container Emplacement Part 1

```
#!python
##Cubit v15.3, Linux
cubit.cmd('reset')
error_old = cubit.get_error_count()
import cPickle
import math, time

#################################################################
# Create first grouping of Pipe Overpack Containers
# For rockfall mesh shift each additional level of stacked
# drums 1 inch to the right of the previous lower level of drums
#################################################################

# Define conversion factors
in_m = 0.0254

#Define the output mesh name
mesh_name = 'detailed_pop_row_rockfall_part1'

#Define the single canister POP cub file name
pop_cub_name = 'detailed_pop_2019_01_19.cub\''

#Import the POP
cubit.cmd('import cubit \'' + pop_cub_name + ' lite unique_genesis_ids')

#Copy the POP to make a 7-pack
theta_list = [x*math.pi/3.0 for x in range(6)]
#Define the center-to-center distance between POPs
gap = 5.0 / 1000.0
pop_g = {}
d_t = 1.50 / 1000
pop_g['D'] = (22.50 * in_m) + 2.0*d_t
pop_g['L'] = (34.25 - 2.0*0.87) * in_m
R = pop_g['D'] + gap

#Define the misalignment in the x direction
#Assume an inch
xdir_adjust = 1 * in_m

vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
for theta in theta_list:
   factor = 1.
   dx = [factor * R * math.cos(theta), factor* R * math.sin(theta), 0.0]
   cubit.cmd('volume ' + vol_list_str + ' copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

#Calculate overall dimensions of the 7-pack
g = dict()
g['L_y'] = 2.0 * (pop_g['D'] + gap) * math.cos(math.pi/6.0) + pop_g['D']  + gap # Added gap JLB
g['L_x'] = g['L_y'] / math.cos(math.pi/6.0) # JLB
g['L_z'] = pop_g['L'] * 3.0
#Calculate the face length
g['F'] = g['L_y'] * math.tan(math.pi/6.0)
#Calculate the depth of the V
g['V'] = g['F'] * math.sin(math.pi/6.0)

#Move the first 7-pack into position
vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
cubit.cmd('group "pop_stack_assembly" add volume all')

dx = [(g['L_x'] - g['V']), i_y * g['L_y'] / 2.0, 0.0]

Assemble into a row of POPs
for i in range(1,3):
  print i
  i_y = 0.0
  dx = [(g['L_x'] - g['V']) * (2*i), i_y * g['L_y'] / 2.0, 0.0]
  print dx
  cubit.cmd('volume ' + vol_list_str + ' copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

#Stack the row of POPs to make them 3 tall
vol_list = cubit.parse_cubit_list('volume', 'all')
```

```
vol_list_str = str(vol_list)[1:-1]
i_list = [-1, 1]

# This puts a drum row below and then another above the original
for i in i_list:
    dx_add = xdir_adjust * i
    dx = [0.0 + dx_add, 0.0, float(i) * pop_g['L']]
    cubit.cmd('volume ' + vol_list_str + ' copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

## Move Row of POP's so that the assembly is centered at 0 0 0
cubit.cmd('group "pop_row_assembly" add volume all')
cubit.cmd('move group with name "pop_row_assembly" midpoint location 0.0 0.0 0.0' )
dx = [(g['L_x'] - g['V']), 0.0, 0.0]
cubit.cmd('move group with name "pop_row_assembly" midpoint location %r %r %r' %(dx[0], dx[1], dx[2]))
#
#
###  === Create Blocks  ===  ###
cubit.cmd('block 100 surface in volume with name "V_D_W*"')
cubit.cmd('block 100 name "pop_drum_wall"')
#
cubit.cmd('block 101 surface in volume with name "V_D_B*"')
cubit.cmd('block 101 name "pop_drum_base"')
#
cubit.cmd('block 102 surface in volume with name "V_D_L*"')
cubit.cmd('block 102 name "pop_drum_lid"')
#
cubit.cmd('block 200 volume with name "V_Base_Ply*"')
cubit.cmd('block 200 name "pop_plywood_lower"')
#
cubit.cmd('block 201 volume with name "V_Lid_Ply*"')
cubit.cmd('block 201 name "pop_plywood_upper"')
#
cubit.cmd('block 202 volume with name "V_Base_Fib*"')
cubit.cmd('block 202 name "pop_fiberboard_lower"')
#
cubit.cmd('block 203 volume with name "V_Lid_Fib*"')
cubit.cmd('block 203 name "pop_fiberboard_upper"')
#
cubit.cmd('block 204 volume with name "V_Side_Fib*"')
cubit.cmd('block 204 name "pop_fiberboard_side"')

cubit.cmd('block 205 volume with name "V_Flange_Fib*"')
cubit.cmd('block 205 name "pop_fiberboard_flange"')

cubit.cmd('block 300 surface in volume with name "V_Pipe_W*"')
cubit.cmd('block 300 name "pop_cv_wall"')
#
cubit.cmd('block 301 surface in volume with name "V_Pipe_B*"')
cubit.cmd('block 301 name "pop_cv_base"')
#
cubit.cmd('block 302 volume with name "V_Pipe_L*"')
cubit.cmd('block 302 name "pop_cv_top_flange"')

#ndrums and inc are problem specific as are the integer values used
# in the ns1,ns3,ns5,ns7 definitions
ndrums = 63
inc = 144
for i in range(1,ndrums):
   ns1 = 457 + (i - 1)*inc
   ns2 = ns1 + 1
   ns3 = 494 + (i - 1)*inc
   ns4 = ns3 + 1
   ns5 = 530 + (i - 1)*inc
   ns6 = ns5 + 1
   ns7 = 560 + (i - 1)*inc
   ns8 = ns7 + 1
   surf_list = ns1,ns2,ns3,ns4,ns5,ns6,ns7,ns8
   surf_list_str = str(surf_list)[1:-1]
   cubit.cmd('nodeset ' + str(i+1) + ' add surface '+ surf_list_str)
   cubit.cmd('nodeset ' + str(i+1) + ' name ' + '"tracker_ns_' + str(i+1) +'"')

#Export the Mesh
import os
pwd = os.getcwd()
export_path = os.path.join(pwd, mesh_name + '.g')
print export_path
cubit.cmd('export genesis \'%s\' overwrite' % export_path)

#Save the geometry values
import cPickle
#cPickle can save as a binary or a text file.  Binaries are smaller, so I
#use them here.  The 'wb' option says to open the binary file with write access.
geom_path = os.path.join(pwd, mesh_name.rsplit('_',4)[0] + '-geom_values_2019_01_19_rockfall_part1.pkl')
geom_file = open(geom_path,'wb')
#Save the file.  Protocol 2 is the faster binary protocol for Python 2.7
cPickle.dump(g, geom_file, protocol = 2)
geom_file.close()

#Print the error count
error_new = cubit.get_error_count()
print 'Cubit error count = %r' %(error_new - error_old)
```

Container Emplacement Part 2

```
#!python
#Cubit v15.3, Linux
cubit.cmd('reset')
error_old = cubit.get_error_count()
import cPickle
import math, time

#################################################################
# Create second grouping of Pipe Overpack Containers
# For rockfall mesh shift each additional level of stacked
# drums 1 inch to the right of the previous lower level of drums
#################################################################

# Define conversion factors
in_m = 0.0254

#Define the output mesh name
mesh_name = 'detailed_pop_row_rockfall_part2'

#Define the single canister POP cub file name
pop_cub_name = 'detailed_pop_2019_01_19.cub\''

#Import the POP
cubit.cmd('import cubit \'' + pop_cub_name + ' lite unique_genesis_ids')

#Copy the POP to make a 7-pack
theta_list = [x*math.pi/3.0 for x in range(6)]
#Define the center-to-center distance between POPs
gap = 5.0 / 1000.0
pop_g = {}
d_t = 1.50 / 1000
pop_g['D'] = (22.50 * in_m) + 2.0*d_t
pop_g['L'] = (34.25 - 2.0*0.87) * in_m
R = pop_g['D'] + gap

#Define the misalignment in the x direction
#Assume an inch
xdir_adjust = 1 * in_m
```

```
vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
for theta in theta_list:
   factor = 1.
   dx = [factor  * R * math.cos(theta), factor * R * math.sin(theta), 0.0]
   cubit.cmd('volume ' + vol_list_str + ' copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

#Calculate overall dimensions of the 7-pack
g = dict()
g['L_y'] = 2.0 * (pop_g['D'] + gap) * math.cos(math.pi/6.0) + pop_g['D']  + gap # Added gap JLB
g['L_x'] = g['L_y'] / math.cos(math.pi/6.0) # JLB
g['L_z'] = pop_g['L'] * 3.0
#Calculate the face length
g['F'] = g['L_y'] * math.tan(math.pi/6.0)
#Calculate the depth of the V
g['V'] = g['F'] * math.sin(math.pi/6.0)

#Move the first 7-pack into position
vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
cubit.cmd('group "pop_stack_assembly" add volume all')
cubit.cmd('move group with name "pop_stack_assembly" midpoint location 0.0 0.0 0.0')


#Assemble into a row of POPs
for i in range(1,3):
  print i
  i_y = 0.0
  dx = [(g['L_x'] - g['V']) * 2*i, i_y * g['L_y'] / 2.0, 0.0]
  print dx
  cubit.cmd('volume ' + vol_list_str + ' copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

#time.sleep(10)

#Create second row of POPs
dx = [0.0, g['L_y'], 0.0]
cubit.cmd('volume all copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

#Stack the row of POPs to make them 3 tall
vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
i_list = [-1, 1]

# This puts a drum row below and then another above the original
for i in i_list:
    dx_add = xdir_adjust * i
    dx = [0.0 + dx_add, 0.0, float(i) * pop_g['L']]
    cubit.cmd('volume ' + vol_list_str + ' copy move x %r y %r z %r' %(dx[0], dx[1], dx[2]))

# Move Row of POP's so that the assembly is centered at 0 0 0
cubit.cmd('group "pop_row_assembly" add volume all')
cubit.cmd('move group with name "pop_row_assembly" midpoint location 0.0 0.0 0.0')

#
#
###  === Create Blocks  ===  ###
cubit.cmd('block 100 surface in volume with name "V_D_W*"')
cubit.cmd('block 100 name "pop_drum_wall_2"')
cubit.cmd('block 100 renumber 1000')
#
cubit.cmd('block 101 surface in volume with name "V_D_B*"')
cubit.cmd('block 101 name "pop_drum_base_2"')
cubit.cmd('block 101 renumber 1010')
#
cubit.cmd('block 102 surface in volume with name "V_D_L*"')
cubit.cmd('block 102 name "pop_drum_lid_2"')
cubit.cmd('block 102 renumber 1020')
#
```

```
cubit.cmd('block 200 volume with name "V_Base_Ply*"')
cubit.cmd('block 200 name "pop_plywood_lower_2"')
cubit.cmd('block 200 renumber 2000')
#
cubit.cmd('block 201 volume with name "V_Lid_Ply*"')
cubit.cmd('block 201 name "pop_plywood_upper_2"')
cubit.cmd('block 201 renumber 2010')
#
cubit.cmd('block 202 volume with name "V_Base_Fib*"')
cubit.cmd('block 202 name "pop_fiberboard_lower_2"')
cubit.cmd('block 202 renumber 2020')
#
cubit.cmd('block 203 volume with name "V_Lid_Fib*"')
cubit.cmd('block 203 name "pop_fiberboard_upper_2"')
cubit.cmd('block 203 renumber 2030')
#
cubit.cmd('block 204 volume with name "V_Side_Fib*"')
cubit.cmd('block 204 name "pop_fiberboard_side_2"')
cubit.cmd('block 204 renumber 2040')

cubit.cmd('block 205 volume with name "V_Flange_Fib*"')
cubit.cmd('block 205 name "pop_fiberboard_flange_2"')
cubit.cmd('block 205 renumber 2050')
#
cubit.cmd('block 300 surface in volume with name "V_Pipe_W*"')
cubit.cmd('block 300 name "pop_cv_wall_2"')
cubit.cmd('block 300 renumber 3000')
#
cubit.cmd('block 301 surface in volume with name "V_Pipe_B*"')
cubit.cmd('block 301 name "pop_cv_base_2"')
cubit.cmd('block 301 renumber 3010')
#
cubit.cmd('block 302 volume with name "V_Pipe_L*"')
cubit.cmd('block 302 name "pop_cv_top_flange_2"')
cubit.cmd('block 302 renumber 3020')

#ndrums and inc are problem specific as are the integer values used
# in the ns1,ns3,ns5,ns7 definitions

ndrums_part1 = 63
ndrums_part2 =126
#have to handle the first drum individually since the surface ids do not increment like
#the remaining drums
ns1 = 198
ns2 = 292
ns3 = 328
ns4 = 329
ns5 = 371
ns6 = 372
ns7 = 414
ns8 = 415
surf_list = ns1,ns2,ns3,ns4,ns5,ns6,ns7,ns8
surf_list_str = str(surf_list)[1:-1]
cubit.cmd('nodeset ' + str(ndrums_part1 + 1) + ' add surface '+ surf_list_str)
#cubit.cmd('nodeset ' + str(ndrums_part1 + 1) + ' name ' + '"tracker_ns_' + str(ndrums_part1 + 1) +'"')
#delete nodeset 1 since it is already incorporated in part 1
cubit.cmd('delete nodeset 1')

#now generate the remaining drum nodesets
inc = 144
for i in range(1,ndrums_part2):
   ns1 = 457 + (i - 1)*inc
   ns2 = ns1 + 1
   ns3 = 494 + (i - 1)*inc
   ns4 = ns3 + 1
   ns5 = 530 + (i - 1)*inc
   ns6 = ns5 + 1
   ns7 = 560 + (i - 1)*inc
   ns8 = ns7 + 1
   surf_list = ns1,ns2,ns3,ns4,ns5,ns6,ns7,ns8
   surf_list_str = str(surf_list)[1:-1]
   cubit.cmd('nodeset ' + str(ndrums_part1 + i + 1) + ' add surface '+ surf_list_str)

# Remove volumes outside of symmetry planes
cubit.cmd('delete volume with y_coord < %r' %(-2*pop_g['D']))
cubit.cmd('delete volume with y_coord > %r' %( 2*pop_g['D']))

# Remove the nodesets associated with volumes that were just deleted
nstart = ndrums_part1 + 1
nend   = ndrums_part1 + ndrums_part2 + 1
for nodeset_id in range(nstart,nend):
    nodeset_list = cubit.get_nodeset_nodes_inclusive(nodeset_id)
    size_list = len(nodeset_list)
    if size_list == 0:
       cubit.cmd('delete nodeset %r' %(nodeset_id))

# Now renumber the nodesets with consecutively numbered names
nodeset_id_list = cubit.get_nodeset_id_list()
size_list = len(nodeset_id_list)
for i in range(size_list):
    cubit.cmd('nodeset ' + str(nodeset_id_list[i]) + ' name ' + '"tracker_ns_' + str(ndrums_part1 + i + 1)
    ↪  +'"')


cubit.cmd('nodeset 1000 add node with y_coord >= ' + str(g['L_y']/2.0 - 0.0005) \
                          + 'and with y_coord <= ' + str(g['L_y']/2.0 + 0.0005) \
                          + 'in volume with is_sheetbody')
cubit.cmd('nodeset 1000 name "ns_sym_shell_pos_y"')

cubit.cmd('nodeset 1001 add node with y_coord <= ' + str(-(g['L_y']/2.0 - 0.0005)) \
                          + 'and with y_coord >= ' + str(-(g['L_y']/2.0 + 0.0005)) \
                          + 'in volume with is_sheetbody')
cubit.cmd('nodeset 1001 name "ns_sym_shell_neg_y"')

cubit.cmd('nodeset 2000 add node with y_coord >= ' + str(g['L_y']/2.0 - 0.0005) \
                          + 'and with y_coord <= ' + str(g['L_y']/2.0 + 0.0005) \
                          + 'in volume with not is_sheetbody')

cubit.cmd('nodeset 2000 name "ns_sym_hex_pos_y"')

cubit.cmd('nodeset 2001 add node with y_coord <= ' + str(-(g['L_y']/2.0 - 0.0005)) \
                          + 'and with y_coord >= ' + str(-(g['L_y']/2.0 + 0.0005)) \
                          + 'in volume with not is_sheetbody')
cubit.cmd('nodeset 2001 name "ns_sym_hex_neg_y"')

#cubit.cmd('delete block 100 101 102 200 201 300 301 302 303')

#Export the Mesh
import os
pwd = os.getcwd()
export_path = os.path.join(pwd, mesh_name + '.g')
print export_path
cubit.cmd('export genesis \'%s\' overwrite' % export_path)

#Save the geometry values
import cPickle
#cPickle can save as a binary or a text file.  Binaries are smaller, so I
#use them here.  The 'wb' option says to open the binary file with write access.
geom_path = os.path.join(pwd, mesh_name.rsplit('_',4)[0] + '-geom_values_2019_01_19_rockfall_part2.pkl')
geom_file = open(geom_path,'wb')
#Save the file.  Protocol 2 is the faster binary protocol for Python 2.7
cPickle.dump(g, geom_file, protocol = 2)
geom_file.close()

#Print the error count
error_new = cubit.get_error_count()
```

```
print 'Cubit error count = %r' %(error_new - error_old)
```

Combine Containers and Disposal Room

```
#!python
#Cubit v15.2
cubit.cmd('reset')
error_old = cubit.get_error_count()
import math
import cPickle

#Define the output mesh name
mesh_name = 'disposal_room_pop_assembly_lowerHorizon_rockFall'

#Define the POP and disposal room names
#pop = pipe over pack
pop_name = 'detailed_pop'
pop_mesh_name_1 = 'detailed_pop_row_rockfall_part1'
pop_mesh_name_2 = 'detailed_pop_row_rockfall_part2'

#dr = disposal room
dr_mesh_name =  'disposal_room-with_clay_slip-lower_horizon-rockFall'

def load_pickled_file(path):
    #Load the saved data (the 'r' tells open() to open as
    #read only, and the 'b' tells it to open as a binary and ignore '\n' - type
    #characters since they only apply to text files.)
    f = open(path, 'rb')
    out = cPickle.load(f)
    f.close()
    return(out)

#Load the geometry values for the POP and disposal room
pop_g = load_pickled_file('detailed-geom_values_2019_01_19_rockfall_part1.pkl')
dr_g  = load_pickled_file('disposal_room-with_clay_slip-lower_horizon-rockFall.pkl')


#Import the POP(s)
cubit.cmd('import mesh geometry \'' + pop_mesh_name_1 + '.g\' unique genesis ids')
cubit.cmd('import mesh geometry \'' + pop_mesh_name_2 + '.g\' unique genesis ids')

#Move the POP to the disposal room
pop_vol_list = cubit.parse_cubit_list('volume', 'all')
pop_vol_list_str = str(pop_vol_list)[1:-1]
cubit.cmd('group "pop_row_assembly" add volume all')
cubit.cmd('move group with name "pop_row_assembly" midpoint location 0.0 0.0 0.0')

dx = [ 0.0, 0.0, dr_g['room_bot_elev'] + pop_g['L_z']/2.0]
cubit.cmd('move volume ' + pop_vol_list_str + ' x %r y %r z %r include_merged' %(dx[0], dx[1], dx[2]))

#Import the disposal room
cubit.cmd('import mesh geometry \'' + dr_mesh_name + '.g\' unique genesis ids')

#Export the Mesh
import os
pwd = os.getcwd()
export_path = os.path.join(pwd, mesh_name + '.g')
print export_path
cubit.cmd('export genesis \'%s\' overwrite' % export_path)

#Print the error count
error_new = cubit.get_error_count()
print 'Cubit error count = %r' %(error_new - error_old)
```

A.4.2. Sierra Input Files

Gradual Compaction Simulation

```
begin sierra WIPP Isothermal Disposal Room

   title Simulation of WIPP Isothermal Disposal Room with POP disposal canisters
   # THis model used the low disposal room horizon and flex foam model for the
   # Celotex fiberboard.

   restart = automatic

   #{KelFactor = 273.15} # convert from C to K
   #{tempK     = 300} #degK

   # Celotex tensile strength (Pa)
   #{psi_to_Pa = 6894.7573} # psi_to_Pa
   #{ten_str_psi = 150} psi
   #{tensile_strength = ten_str_psi * psi_to_Pa} Pa


   #Define the number of drums in the simulation
   ${ndrums = 153}

   #Define the scale factors
   ${s_1_0 = 2.0e-2}
   ${s_ramp = 2e9}

   # =========================== Directions ========================== #

   define direction x with vector 1.0 0.0 0.0
   define direction y with vector 0.0 1.0 0.0
   define direction z with vector 0.0 0.0 1.0
   define direction negative_z with vector 0.0  0.0 -1.0
   define point origin with coordinates 0.0 0.0 0.0


   # =========================== Functions ========================== #
   begin definition for function function_constant
      type is piecewise linear
      begin values
         -1.0e16  1.0
         1.0e16  1.0
      end
   end

   begin definition for function fluid_density_ramp
      #Specify the density to give the appropriate lithostatic pressure at room at start of simulation and then
      ↪  ramp it down to zero
      type is analytic
      evaluate expression is "2300.0 / 2.0 * (1.0 - cos(pi * time / 0.5))"
   end definition for function fluid_density_ramp


   #---------- Viscoplastic Rate Scaling Functions ---------

   # Sierra/SM calculates user variables at the beginning and the end of any time step where data is
   # written to the exodus database.  Normally this is fine, but if variable x is an input and an
   # output then x would get updated twice if data were written to the exodus database instead of
   # once in a time step.  To make sure x is only calculated at the beginning (or end) of the time
   # step we compare the global variable time against a user variable called stored_time.  The global
   # variable time is always the time at the end of the time step.  (This is a bit weird since all
   # other variables can change from the start to the end of the time step, but one can always get
   # the time at the beginning of a step by calculating t-dt.)  As long as we update stored_time
   # AFTER x, then the beginning of a time step corresponds to time != stored_time, and the end of
   # the time step corresponds to time == stored_time.  If data is not written to the exodus database,
   # then I think user variables are calculated only at the beginning of a time step.

   begin function stored_time_function
      type is analytic
      expression variable: t_stored = global time
      evaluate expression = "t_stored"
   end

   begin function vp_rate_scale_factor_function
      #The start time column specifies when the corresponding temperature expression starts to be used.
      #(We have to add a small amount to start times to make sure the last time step
      #of the previous time period does not use the temperature expression for the next time period.)
      type is piecewise analytic
      begin expressions
         -0.5                    "0.02"
         1e-10             "0.02 + 2000000000 * (time - 0)"
      end expressions
   end

   begin function vp_rate_scale_factor_broadcast_function
      type is analytic
      expression variable: vp_rate_scale_factor = global vp_rate_scale_factor
      evaluate expression = "vp_rate_scale_factor"
   end

   begin function mat_time_function
      type is analytic
      expression variable: t = global time
      expression variable: dt = global timestep
      expression variable: t_stored = global stored_time
      expression variable: t_m = global mat_time
      expression variable: s = global vp_rate_scale_factor
      evaluate expression = "                                 \#
         (t != t_stored) ? (                                    \#
         t_m + s * dt                                       \#
         ) : (                                                  \#
         t_m                                                \#
         )                                                      \#
      "
   end

   # =========================== Materials ========================== #

   begin property specification for material halite

      density = 2300.0

      begin parameters for model md_viscoplastic

         #Intact salt parameters
         #(These Cal 3B parameters are from Table 5.2 of SAND2018-12601.)
         bulk modulus  = 20.7e9 # Pa  compute bulk modulus from E=31e+09 Pa and nu = 0.25
         shear modulus = 12.4e9  # Pa
         a0            = 5.617e1 # 1/sec
         q0/R          = 5123 # K
         n0            = 1.595
         a1            = 8.386e22 # 1/sec
         q1/R          = 12580.5 # K = 25000 cal / mol / 1.9872035 cal / (mol K)
         n1            = 5.5
         b1            = 6.086e6 # 1/sec
         a2            = 4.415e16 # 1/sec
         q2/R          = 5123 # K
         n2            = 6.279
         b2            = 3.034e-2 # 1/sec
         sigma_g       = 20570000.0 # Pa
         q             = 5335.0
         m0            = 0.9201
         k0            = 5.277e-2
         c0            = 8.882e-3 # 1/K
         m1            = 5.282
         k1            = 3.052e12
         c1            = 8.882e-3 # 1/K
         alpha_h       = 3.367
         beta_h        = -0.6838
         alpha_r       = 0.58
         a             = 16.0
         alpha         = 45.0e-6

      end parameters for model md_viscoplastic

   end property specification for material halite

   begin property specification for material anhydrite
      density = 2300.0 # kg / m^3
      #Sierra/SM needs to have elastic parameters defined with these names, so we repeat
      #the values in the "begin parameters ..." block.
      bulk modulus = 83.4444444e9 # Pa = bulk modulus
      shear modulus = 27814814814 # Pa = shear modulus

      #These Drucker-Prager parameters came from SAND97-0795, Table 6, except it does not
      #mention a dilation angle.  They almost certainly used the soil and foam model, which
      #has a dilatation angle of 0 degrees.  Here we have switched to associated flow.
      #The Drucker-Prager yield function is often written as phi = sqrt(J_2) + a * I_1 - C
      #B0 = bulk modulus = 83.4444444e9 Pa
      #G0 = shear modulus = 27814814814 Pa
      #C = 1.35
      #a = 0.45
      #alpha = dilatation angle = 24.227745 deg = atan(0.45)

      #Reduce Kayenta to Drucker-Prager
      begin parameters for model kayenta
         B0 = 83.4444444e9   # Pa = bulk modulus
         G0 = 27814814814    # Pa = shear modulus
         J3TYPE = 1          # Sets the dependence on J_3.  J3TYPE = 1 is a Von-Mises dependence.
         A1 = 1350000        # Pa
         A2 = 0.0
         A3 = 0.0
         A4 = 0.45
         RK = 1.0
         P0 = -1.0e99        # Put the compression cap at virtually infinity
         P1 = 0.0            # No cap
         P2 = 0.0            # No cap
         P3 = 0.0            # Zero porosity
         CR = 0.001          # Minimize the size of the curved part of the cap
         HC = 0.0            # Disable kinematic hardening
         RN = 0.0            # Disable kinematic hardening
      end parameters for model kayenta
   end property specification for material anhydrite

   begin definition for function polyhalite_pressure_volstrain_function
      type is piecewise linear
      ordinate is volumetric_strain
      abscissa is pressure
      begin values
         -1  -6.583333333e+10 # -65.83333333 GPa
         0  0
         1  6.583333333e+10 # 65.83333333 GPa = bulk modulus = E / (3*(1-2*nu)), where E = 55.3 GPa and nu =
         ↪  0.36
      end values
   end definition for function polyhalite_pressure_volstrain_function

   begin property specification for material polyhalite

      #These parameters came from SAND97-0796, Pages A-40 through A-41, Section 2.5.2

      #The yield function for this model is
      #phi = sqrt(3*J_2) - (a0 + a1 * p + a2 * p^2)
      #This model has a Drucker-Prager yield surface if a2 = 0
      #The Drucker-Prager yield function is often written as
      #phi = sqrt(J_2) + a * I_1 - C
      #Thus, we can write a0 and a1 in terms of a and C

      density = 2300.0
      begin parameters for model soil_foam
         poissons ratio  = 0.36
         youngs modulus  = 5.53e+10 # Pa
         a0              = 2459512.147 # Pa = sqrt(3) * C, where C = 1.42 MPa
         a1              = 2.457780096 # unitless = 3 * sqrt(3) * a, where a = 0.473
         a2              = 0.0
         pressure cutoff = -1000000.0 # Pa
         pressure function = polyhalite_pressure_volstrain_function
      end parameters for model soil_foam
   end property specification for material polyhalite

   ##########################
   ## 304L Stainless Steel ##
   ##########################
   #
   # 18Cr-8Ni, 304L Stainless Steel Plate Per ASTM A240 (SA-240), UNS S30403
   #

   begin property specification for material SA-240_304L

      density = 7916.5 # MMPDS-01 Table 2.7.1.0(b)

      begin parameters for model ml_ep_fail
         youngs modulus = 2.02547e+11 # MMPDS-01 Table 2.7.1.0(b), Average of Tension and Compression Moduli
          ↪  and then inadvertently scaled by 1.03
         poissons ratio = 0.27 # MMPDS-01 Table 2.7.1.0(b)
         yield stress = 2.77e8 #First point on hardening curve
         beta = 1.0
         hardening function = SA-240_304L_Hardening_Func_20
         critical tearing parameter = 1.64 # Consistent with last plastic strain point in the hardening curve
         critical crack opening strain = 0.04828 # Engineering judgement
      end

   end

   # Hardening functions based on data obtained from ASME SA-240M Table 2(Ftu, Fty, e), INL/CON-06-11962
   # (Blandford, R. K., Morton, D. K., Snow, S. D., and Rahl, T. E., Tensile Stress-Strain Results for
   # 304L and 316L Stainless Steel Plate at Temperature, Idaho National Laboratory, 2007)

   # Curve supplied by Ben Reedlunn based on data in the Blandford paper referenced above
   begin definition for function SA-240_304L_Hardening_Func_20
      type = piecewise linear
      abscissa = plastic_strain
      ordinate = yield_stress
      begin values
         0.00000   2.77E+08
         4.73E-03  3.08E+08
         1.09E-02  3.31E+08
         2.32E-02  3.66E+08
         4.79E-02  4.23E+08
         9.74E-02  5.23E+08
         1.47E-01  6.14E+08
         1.97E-01  6.92E+08
         2.46E-01  7.65E+08
         2.96E-01  8.34E+08
         3.46E-01  9.00E+08
         3.95E-01  9.66E+08
         4.45E-01  1.03E+09
         4.95E-01  1.09E+09
         1.64E+00  2.30E+09
      end
```

```
end
#
#
# +-+-+- ASTM A1008 Drum Steel +-+-+- #

begin material ASTM_A1008_Drum_Steel

   density = 7750.0 # ASME B&PV Code 2015, Section II Part D, Table PRD

   begin parameters for model ml_ep_fail
      youngs modulus = 2.02000e+11 # ASME B&PV Code 2015, Section II Part D, Table TM-1 (Carbon steels with
      ↪  C <= 0.30%)
      poissons ratio = 0.300 # ASME B&PV Code 2015, Section II Part D, Table PRD
      yield stress = 2.7400e+08 # WIPP Drum Lid Material Test Data
      ↪  (WIPP_Drum_Material_Test_Data_2017_01_12.xlsx)
      beta = 1.0
      hardening function = ASTM_A1008_Drum_Steel_Hardening_Func_293
      critical tearing parameter = 1.635 # Based on Reduction in Area from WIPP Drum Lid Material Test Data
      ↪  (WIPP_Drum_Material_Test_Data_2017_01_12.xlsx)
      critical crack opening strain = 0.01257 # Engineering judgement
   end

end

# Hardening functions based on data obtained from:
#  WIPP Drum Lid Material Test Data (WIPP_Drum_Material_Test_Data_2017_01_12.xlsx) --> Shape of Stress
↪  Strain Curve, Fty, Ftu

begin function ASTM_A1008_Drum_Steel_Hardening_Func_293
   type = piecewise linear
   abscissa = plastic_strain
   ordinate = yield_stress
   begin values
      0.0000  2.7400e+08
      0.0150  2.7652e+08
      0.0250  2.9559e+08
      0.0350  3.1039e+08
      0.0450  3.2203e+08
      0.0500  3.2702e+08
      0.1000  3.6250e+08
      0.1500  3.8746e+08
      0.2000  4.0771e+08
      0.2500  4.2443e+08
      1.6340  8.4663e+08
   end
end
#
#
# +-+-+- Plywood +-+-+- #
#

begin material Plywood

   # These paramters were provided by Ben Reedunn based on the paper
   #
   # "Relationships between small-specimen and large panel bending tests on structural
   # wood-based panels" by J. Dobbin McNatt and Robert W. Wellwood and Lars Bach
   # Forest Products Journal, 1990, Vol 40, Num 9, pp 10-16
   #
   # The mas stress was determined as the average of the parallel and perpendicular
   # strengths, and then it was scaled down by 20 % since wood will fail at a constant
   # load if held for long enough time, as discussed in
   #
   # "Duration of Load Revisited" by Preben Hoffmeyer and John Sorensen.
   # Wood Science and Technolology, December 2007, Vol 41, Issue 8, pp 687-711.

   density = 576.7
   begin parameters for model elastic_fracture
      youngs modulus = 5.30e+09 # Pa
      poissons ratio = 0.35 # unitless
      max stress     = 5.00e+06 # Pa
      critical crack opening strain = 0.001 # engineering judgement
   end parameters for model elastic_fracture

end



begin material fiberboard

   # The elastic plastic parameters were provided by Jim Bean based on the SAND report
   #
   # SAND2004-1390 by Byoung Park and Frank Hansen "Simulations of the Pipe Overpack Compute Constitutive
   ↪  Model Parameters for Use in WIPP Room Closure Calculations"
   # See Table 2
   density = 256.0 #kg/m^3

   begin parameters for model elastic_plastic
      youngs modulus = 6.895e7 # Pa
      poissons ratio = 0.05    # m/m
      yield stress = 3.447e5   # Pa
      hardening modulus = 2.413e7 # Pa
      beta = 1.0 # unitless changed from reference value of 0.5 (no idea where they got that value)
   end

   #Flex foam parameters were taken from the SNL report SAND2018-2433 by M. Neilsen, Wei-Yang Lu, Brian
   ↪  Werner, William Scherzinger, and David Lo "Flexible Foam Model" for a 15pcf polyurethane foam.
   #The paramterized Model is a "rough match" to the room temp celotex load deflection curves from the Oak
   ↪  Ridge Report Y/EN-4120 "Packaging Materials Properties Data" by M.S Walker (Jan. 1991)

   #Poissons ratio was set to zero based on a staement in a report from Westinghouse Savannah RIver company
   ↪  by A.C. Smith and P.R. Vormelker "Celotex Structural Properties Test (WSRC-TR-2000-00444)
      #"During testing, it was noted that lateral expansion(bulging) of the specimens did not occur".

   #The WSRC report discusses tensile test on dogbone and multilayer samples.  The data show a wide range of
   ↪  tensile strengths depending on factors such as relative humidity, temperature and
   #whether loading is parallel or perpindicular to the celotex sheets. When multiple sheets are glued
   ↪  together the glue strength plays a large role (reduced strength) when loading is perpindicular to the
   ↪  sheets.

   begin parameters for model flex_foam
      youngs modulus           = {120000.0* psi_to_Pa}  #Pa
      poissons ratio           = 0.0000 #0.250
      phi                      = 0.200
      flow rate                = 1.000
      power exponent           = 1.000
      dev multiplier           = 0.2
      tensile strength         = {tensile_strength} #Pa
      adam                     = 1.0
      bdam                     = 0.5
      youngs function          = foam_yModulus
      poissons function        = foam_Constant
      youngs phi function      = foam_E
      poissons phi function    = foam_Constant
      rate function            = foam_Rate
      exponent function        = foam_Expo
      shear hardening function = foam_Shear
      hydro hardening function = foam_Hydro
      beta function            = foam_Beta
      dmod function            = foam_Modulus
      dpr function             = foam_Constant
      dmod phi function        = foam_E
      dpr phi function         = foam_Constant
      damage function          = foam_Damage
   end parameters for model flex_foam
```

```
end
##
## currently no damage = no failure of foam in tension
##

begin definition for function foam_Damage
  type is piecewise linear
  begin values
     0.00000     0.00000
     0.14000     0.00000
     0.40000     0.00000
   100.00000     0.00000
  end values
end definition for function foam_Damage

begin definition for function foam_Beta
  type is piecewise linear
   begin values
      0.000         0.600
      0.200         0.600
      0.225         0.600
      0.250         0.595
      0.275         0.590
      0.300         0.575
      0.325         0.540
      0.350         0.460
      0.375         0.380
      0.400         0.300
      0.425         0.220
      0.450         0.140
      0.475         0.100
      1.000         0.100
     10.000         0.100
   end values
end definition for function foam_Beta

begin definition for function foam_E
   type is piecewise linear
    begin values
      0.00         0.020
      0.20         0.020
      0.25         0.018
      0.30         0.016
      0.35         0.028
      0.40         0.078
      0.45         0.148
      0.50         0.238
      0.55         0.348
      0.60         0.478
      0.65         0.628
      0.70         0.798
      0.80         1.050
      0.90         1.400
      1.00         1.780
      1.50         3.000
      2.00         6.000
     10.00        10.000
    end values
end

##
## currently just using 60 ppm/C based on measured rigid urethane foam
## flexible should probably be higher.
##

begin definition for function foam_Thermal
     type is piecewise linear
     ordinate is strain
     abscissa is temperature
     begin values
      {-500.0 + KelFactor}  -0.0300
      {   0.0 + KelFactor}   0.0000
      { 500.0 + KelFactor}   0.0300
     end values
end definition for function foam_Thermal

begin definition for function foam_yModulus
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
      {-53.90 + KelFactor}      0.060
      {-40.00 + KelFactor}      0.050
      {-20.00 + KelFactor}      0.030
      {  0.00 + KelFactor}      0.021
      { 21.10 + KelFactor}      0.018
      { 73.90 + KelFactor}      0.015
    end values
end definition for function foam_yModulus


begin definition for function foam_Modulus
  type is piecewise linear
   begin values
       {-53.90 + KelFactor}       6.000
       {-40.00 + KelFactor}       1.000
       {-20.00 + KelFactor}       0.080
       {  0.00 + KelFactor}       0.048
       { 21.10 + KelFactor}       0.036
       { 73.90 + KelFactor}       0.033
   end values
end definition for function foam_Modulus

begin definition for function foam_Constant
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
       {-53.90 + KelFactor}      1.0
       { 21.10 + KelFactor}      1.0
       { 73.90 + KelFactor}      1.0
    end values
end definition for function foam_Constant

begin definition for function foam_Rate
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
       {-53.90 + KelFactor}     -15.00
       { 21.10 + KelFactor}      4.80
       { 73.90 + KelFactor}      8.80
    end values
end definition for function foam_Rate

begin definition for function foam_Expo
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
       {-53.90 + KelFactor}      8.5
       { 21.10 + KelFactor}      5.0
       { 73.90 + KelFactor}      3.0
    end values
end definition for function foam_Expo
```

```
begin definition for function foam_Shear
    type is piecewise linear
    begin values
       0.000      {120.0 * psi_to_Pa}
       0.400      {120.0 * psi_to_Pa}
       0.450      {130.0 * psi_to_Pa}
       0.500      {180.0 * psi_to_Pa}
       0.550      {230.0 * psi_to_Pa}
       1.000      {800.0 * psi_to_Pa}
      10.000     {1000.0 * psi_to_Pa}
    end values
end definition for function foam_Shear

begin definition for function foam_Hydro
    type is piecewise linear
    begin values
       0.000      {100.0 * psi_to_Pa}
       0.400      {100.0 * psi_to_Pa}
       0.450      {130.0 * psi_to_Pa}
       0.500      {200.0 * psi_to_Pa}
       0.550      {250.0 * psi_to_Pa}
       1.000     {1000.0 * psi_to_Pa}
      10.000     {2000.0 * psi_to_Pa}
    end values
end definition for function foam_Hydro


   ## ~~~~~~~~~~~~~~~~~~ Section Definitions ~~~~~~~~~~~~~~~~~~ ##

   begin solid section solid_1
      formulation = selective_deviatoric
      #formulation = mean_quadrature
      strain incrementation = strongly_objective
   end solid section solid_1

   #wall of pipe
   begin shell section shell_t6p35mm #
      thickness = 0.00635# 0.25 in.
      lofting factor = 0.5
   end

   #bottom of pipe
   begin shell section shell_t7p112mm #
      thickness = 0.007112# 0.28 in.
      lofting factor = 0.5
   end

   #drum
   begin shell section shell_t1p5mm
      thickness = 0.0015
      lofting factor = 0.5
   end


   ## ~~~~~~~~~~~~~~~~~~ Finite Element Model ~~~~~~~~~~~~~~~~~~ ##

   begin finite element model room

      database name = mesh/disposal_room_pop_assembly_lowerHorizon.g
      database type = exodusII

      begin block defaults
         effective moduli model = elastic
      end

      #
      begin parameters for block b_halite
         material halite
```

```
      solid mechanics use model md_viscoplastic
      section = solid_1
   end
   #
   begin parameters for block b_arg_halite
      #material arg_halite
      material halite
      solid mechanics use model md_viscoplastic
      section = solid_1
   end
   #
   begin parameters for block b_anhydrite
      material anhydrite
      solid mechanics use model kayenta
      section = solid_1
   end
   #
   begin parameters for block b_polyhalite
      material polyhalite
      solid mechanics use model soil_foam
      section = solid_1
   end
   #
   begin parameters for block \#
      pop_drum_wall   pop_drum_base   pop_drum_lid \#
      pop_drum_wall_2 pop_drum_base_2 pop_drum_lid_2
      material ASTM_A1008_Drum_Steel
      solid mechanics use model ml_ep_fail
      section = shell_t1p5mm
   end

   begin parameters for block pop_cv_top_flange pop_cv_top_flange_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      section = solid_1
   end

   begin parameters for block pop_cv_wall pop_cv_wall_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      section = shell_t6p35mm
   end

   begin parameters for block pop_cv_base pop_cv_base_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      section = shell_t7p112mm
   end

   begin parameters for block \#
      pop_plywood_lower   pop_plywood_upper \#
      pop_plywood_lower_2 pop_plywood_upper_2
      material Plywood
      solid mechanics use model elastic_fracture
      section = solid_1
   end

   begin parameters for block \#
      pop_fiberboard_lower   pop_fiberboard_upper   pop_fiberboard_side   pop_fiberboard_flange \#
      pop_fiberboard_lower_2 pop_fiberboard_upper_2 pop_fiberboard_side_2 pop_fiberboard_flange_2
      material fiberboard
      #solid mechanics use model elastic_plastic
      solid mechanics use model flex_foam
      section = solid_1
   end

end
```

```
## ~~~~~~~~~~~~~~~~~~ Time Step Control ~~~~~~~~~~~~~~~~~~ ##

begin presto procedure the_procedure.1

   begin time control

      begin time stepping block p0
         start time = -0.5
         begin parameters for presto region SmRegion.1
            time step scale factor = 1
            time step increase factor = 5
         end
      end

      begin time stepping block p1
         start time = 0
         begin parameters for presto region SmRegion.1
            time step scale factor = 1
            time step increase factor = 5
         end
      end

      termination time = 1000

   end


   ## ~~~~~~~~~~~~~~~~~~ Begin Analysis ~~~~~~~~~~~~~~~~~~ ##

   begin presto region SmRegion.1

      begin solution termination
         terminate global mat_time > 3.15576e+10
      end

      use finite element model room


      ## ~~~~~~~~~~~~~~~~~~ Mass time  ~~~~~~~~~~~~~~~~~~ ##

      begin mass scaling
         # include all blocks
         # scale to get critical time step close to the minimum of the geomaterials
         block = pop_cv_top_flange   pop_plywood_upper   pop_plywood_lower  \#
                 pop_cv_top_flange_2 pop_plywood_upper_2 pop_plywood_lower_2
         target time step = 4.0e-06
      end

      begin node based time step parameters nbased_tstep
      end


      ## ~~~~~~~~~~~~~~~~~~ Analytic Objects ~~~~~~~~~~~~~~~~~~ ##

      begin analytic object pos_y_sym_analytic_plane
         type = plane
         normal direction = 0.0 -1.0 0.0
         origin = 0.0 1.07886 0.0
         scale factor = 0.5
      end

      begin analytic object neg_y_sym_analytic_plane
         type = plane
         normal direction = 0.0 1.0 0.0
         origin = 0.0 -1.07886 0.0
         scale factor = 0.5
      end
```

```
## ~~~~~~~~~~~~~~~~~~ Boundary Conditions ~~~~~~~~~~~~~~~~~~ ##

begin gravity
   include all blocks
   gravitational constant = 9.79  # SAND88-2948 used 9.79 m/s^2.  9.79 m/s^2 is close to the local
   ↪  value of gravity at WIPP (9.79483 m/s^2).
   direction = negative_z
   function = function_constant
end

begin prescribed temperature
   include all blocks
   function = function_constant
   scale factor = 300.0
end

begin pressure
   surface = ss_top_surf
   function = function_constant
   scale factor = 13.57e+06
end

begin fixed displacement
   node set = ns_bot_surf
   components = z
end

begin fixed displacement
   node set = ns_left_surf
   components = x
end

begin fixed displacement
   node set = ns_right_surf
   components = x
end

begin fixed displacement
   node set = ns_back_surf
   components = y
end

begin fixed displacement
   node set = ns_front_surf
   components = y
end

begin fluid_pressure
   surface = ss_room_surf
   active periods = p0
   #Specify which direction corresponds to the depth of fluid.
   fluid surface normal = z
   #The depth function is defined relative to the reference point.
   #See the figure in the fluid pressure section of the Sierra/SM manual for more info.
   reference point = origin
   #Set the fluid height, relative to the reference point, to give the appropriate lithostatic
   ↪  pressure at room
   #p1 = -15.97e6 Pa
   #p2 = -13.57e6 Pa
   #z1 = -54.19 m
   #z2 = 52.87 m
   #slope = (p2-p1)/(z2-z1)
   #p = slope*(z-z1)+p1
   #z0 = depth = elevation of fluid surface
   #depth = -p1 / slope + z1
   depth = 658.2050833333333 #m = height
```

```
   #Specify the function to ramp the fluid density down by time = 0
   density function = fluid_density_ramp
   gravitational constant = 9.79
end


## ~~~~~~~~~~~~~~~~~~ CONTACT MDOEL ~~~~~~~~~~~~~~~~~~ ##

begin contact definition contact_defn

   enforcement = al

   begin enforcement options
      momentum balance iterations = 20
   end

   #---------- Contact Surface Definitions ----------

   begin contact surface pop_surf
      block = pop_drum_wall \#
              pop_drum_base \#
              pop_drum_lid \#
              pop_plywood_lower \#
              pop_plywood_upper \#
              pop_cv_wall \#
              pop_cv_base \#
              pop_cv_top_flange \#
              pop_fiberboard_lower \#
              pop_fiberboard_upper \#
              pop_fiberboard_side \#
              pop_fiberboard_flange
   end

   begin contact surface pop_surf_2
      block = pop_drum_wall_2 \#
              pop_drum_base_2 \#
              pop_drum_lid_2 \#
              pop_plywood_lower_2 \#
              pop_plywood_upper_2 \#
              pop_cv_wall_2 \#
              pop_cv_base_2 \#
              pop_cv_top_flange_2 \#
              pop_fiberboard_lower_2 \#
              pop_fiberboard_upper_2 \#
              pop_fiberboard_side_2 \#
              pop_fiberboard_flange_2
   end


   begin contact surface room_surf
      surface = ss_room_surf
   end

   contact surface clay_A_master contains SS_clay_A_master
   contact surface clay_A_slave contains SS_clay_A_slave

   contact surface clay_B_master contains SS_clay_B_master
   contact surface clay_B_slave contains SS_clay_B_slave

   contact surface clay_D_master contains SS_clay_D_master
   contact surface clay_D_slave contains SS_clay_D_slave

   contact surface clay_E_master contains SS_clay_E_master
   contact surface clay_E_slave contains SS_clay_E_slave

   contact surface clay_F_master contains SS_clay_F_master
   contact surface clay_F_slave contains SS_clay_F_slave
```

```
contact surface clay_G_master contains SS_clay_G_master
contact surface clay_G_slave contains SS_clay_G_slave

contact surface clay_H_master contains SS_clay_H_master
contact surface clay_H_slave contains SS_clay_H_slave

contact surface clay_I_master contains SS_clay_I_master
contact surface clay_I_slave contains SS_clay_I_slave

contact surface clay_J_master contains SS_clay_J_master
contact surface clay_J_slave contains SS_clay_J_slave

contact surface clay_K_master contains SS_clay_K_master
contact surface clay_K_slave contains SS_clay_K_slave

contact surface clay_L_master contains SS_clay_L_master
contact surface clay_L_slave contains SS_clay_L_slave

contact surface clay_M_master contains SS_clay_M_master
contact surface clay_M_slave contains SS_clay_M_slave

contact surface pos_y_sym_contact_surface contains pos_y_sym_analytic_plane

contact surface neg_y_sym_contact_surface contains neg_y_sym_analytic_plane


#---------- Friction Model Definitions ----------

begin constant friction model clay_seam_model
   friction coefficient = 0.20
   #friction coefficient = 0.00
end

begin constant friction model salt_to_salt_friction_model
   friction coefficient = 0.50
end

begin constant friction model metal_to_salt_friction_model
   friction coefficient = 0.4 #0.65
end

begin constant friction model metal_to_metal_friction_model
   friction coefficient = 0.30 #0.50
end

begin remove initial overlap
   #          debug iteration plot = ON
   overlap iterations = 10
   overlap normal tolerance     = 1.0e-3
   overlap tangential tolerance = 1.0e-3
end remove initial overlap


#---------- Contact Interaction Definitions ----------

begin interaction defaults
   constraint formulation = face_face
   general contact = on
end

begin interaction clay_A
   master = clay_A_master
   slave = clay_A_slave
   friction model = clay_seam_model
end interaction

begin interaction clay_B
   master = clay_B_master
   slave = clay_B_slave
   friction model = clay_seam_model
end interaction

begin interaction clay_D
   master = clay_D_master
   slave = clay_D_slave
   friction model = clay_seam_model
end

begin interaction clay_E
   master = clay_E_master
   slave = clay_E_slave
   friction model = clay_seam_model
end

begin interaction clay_F
   master = clay_F_master
   slave = clay_F_slave
   friction model = clay_seam_model
end

begin interaction clay_G
   master = clay_G_master
   slave = clay_G_slave
   friction model = clay_seam_model
end

begin interaction clay_H
   master = clay_H_master
   slave = clay_H_slave
   friction model = clay_seam_model
end

begin interaction clay_I
   master = clay_I_master
   slave = clay_I_slave
   friction model = clay_seam_model
end

begin interaction clay_J
   master = clay_J_master
   slave = clay_J_slave
   friction model = clay_seam_model
end

begin interaction clay_K
   master = clay_K_master
   slave = clay_K_slave
   friction model = clay_seam_model
end

begin interaction clay_L
   master = clay_L_master
   slave = clay_L_slave
   friction model = clay_seam_model
end

begin interaction clay_M
   master = clay_M_master
   slave = clay_M_slave
   friction model = clay_seam_model
end interaction

begin interaction salt_to_salt
   surfaces = room_surf room_surf clay_G_master clay_G_slave clay_F_master clay_F_slave
   friction model = salt_to_salt_friction_model
end
```

```
   begin interaction metal_to_salt
      master = room_surf
      slave  = pop_surf
      friction model = metal_to_salt_friction_model
   end

   begin interaction metal_to_salt_2
      master = room_surf
      slave  = pop_surf_2
      friction model = metal_to_salt_friction_model
   end

   begin interaction metal_to_metal
      # self contact
      surfaces = pop_surf pop_surf
      friction model = metal_to_metal_friction_model
   end

   begin interaction metal_to_metal_2
      # self contact
      surfaces = pop_surf_2 pop_surf_2
      friction model = metal_to_metal_friction_model
   end
   begin interaction metal_to_metal_3
      # contact between 2 groups of canister blocks
      surfaces = pop_surf pop_surf_2
      friction model = metal_to_metal_friction_model
   end
   begin interaction metal_to_analytic_plane
      # contact between full drum blocks and plane
      surfaces = pop_surf pop_surf_2 pos_y_sym_contact_surface neg_y_sym_contact_surface
      friction model = metal_to_metal_friction_model
   end

end


## ~~~~~~~~~~~~~~~~~~ Initial Conditions ~~~~~~~~~~~~~~~~~~ ##

begin initial condition
   block = b_halite b_arg_halite b_polyhalite b_anhydrite
   #It is important to initialize unrotated_stress and not stress, because
   #stress is just an output that gets calculated as needed, while
   #unrotated_stress gets stored and used by the constitutive models.
   #As of November 2015, initializing stress does nothing.
   initialize variable name = unrotated_stress
   variable type = element
   subroutine real parameter: top = 52.87
   subroutine real parameter: bot = -54.19
   subroutine real parameter: p1 = -13.57e6
   subroutine real parameter: po = -15.97e6
   subroutine real parameter: kvert_xx = 1.0
   subroutine real parameter: kvert_yy = 1.0
   subroutine real parameter: kvert_zz = 1.0
   subroutine real parameter: kvert_xy = 0.0
   subroutine real parameter: kvert_yz = 0.0
   subroutine real parameter: kvert_zx = 0.0
   subroutine string parameter: dir = Z
   element block subroutine = geo_is
end initial condition

#--------- Viscoplastic Rate Scaling ---------

# See note above stored_time_function.
begin user variable stored_time
   type = global real length = 1
   global operator = min
```

```
            initial value = -0.5
         end

         begin user variable mat_time
            type = global real length = 1
            global operator = min
            #edit the following line for restart runs!
            initial value = -0.01
         end

         begin user variable vp_rate_scale_factor
            type = global real length = 1
            global operator = min
            initial value = 0.02
         end


         ## ~~~~~~~~~~~~~~~~~~ Element Death ~~~~~~~~~~~~~~~~~~ ##
         # Use material criterion to kill wood elements with elastic fracture model
         begin element death pop_wood_failure
            block = pop_plywood_lower   pop_plywood_upper \#
                    pop_plywood_lower_2 pop_plywood_upper_2
            material criterion = on kill when 1 integration points remain
         end

         begin element death pop_drum_failure
            block = pop_drum_wall   pop_drum_base   pop_drum_lid \#
                    pop_drum_wall_2 pop_drum_base_2 pop_drum_lid_2
            material criterion = on kill when 1 integration points remain
         end

         begin element death pop_cv_failure
            block = pop_cv_wall   pop_cv_base   pop_cv_top_flange    \#
                    pop_cv_wall_2 pop_cv_base_2 pop_cv_top_flange_2
            material criterion = on kill when 1 integration points remain
         end

#JEB uncomment when a failure model for the fiberboards is included
#          begin element death pop_fiberboard
#             block = pop_fiberboard_upper   pop_fiberboard_lower   pop_fiberboard_side   pop_fiberboard_flange
↪  \#
#                     pop_fiberboard_upper_2 pop_fiberboard_lower_2 pop_fiberboard_side_2
↪  pop_fiberboard_flange_2
#          end

         begin element death degenerate_element
            include all blocks
            death on inversion = on
            criterion is element value of element_shape <= 0.0
         end


         ## ~~~~~~~~~~~~~~~~~~ User Output Variable Definitions ~~~~~~~~~~~~~~~~~~ ##

         begin user output
            block = b_halite b_arg_halite
            compute global vp_rate_scale_factor as function vp_rate_scale_factor_function
            # Although vp_scale_factor is really a global variable, it needs to be at the integration points
            # so the material model can read it.
            $0
            compute element vp_rate_scale_factor(1) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(2) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(3) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(4) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(5) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(6) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(7) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(8) as function vp_rate_scale_factor_broadcast_function
```

```
   compute at every step
end

begin user output
   compute global mat_time as function mat_time_function
   # See note above stored_time_function.  Note that stored_time must be calculated
   # AFTER other functions that use it.
   compute global stored_time as function stored_time_function
   compute at every step
end


begin user output
   compute element el_avg_stress as average of element cauchy_stress
   compute element el_avg_mean_stress as average of element hydrostatic_stress
   compute element el_avg_eq_stress as average of element eq_stress
   compute element el_avg_eq_tr_strain as average of element eq_tr_strain
   compute element el_avg_eq_vp_strain as average of element eq_vp_strain
   compute element el_avg_log_strain as average of element unrotated_log_strain
   compute element el_avg_vonmises as average of element von_mises
   compute element el_avg_eqps as average of element eqps
end

begin user output
   node set = ns_roof_ctr
   compute global roof_disp as average of nodal displacement
end

begin user output
   node set = ns_floor_ctr
   compute global floor_disp as average of nodal displacement
end

begin user output
   node set = ns_left_wall_ctr
   compute global left_wall_disp as average of nodal displacement
end

begin user output
   node set = ns_right_wall_ctr
   compute global right_wall_disp as average of nodal displacement
end

#These 4 user outputs are for computing average displacements along line of nodes
begin user output
   node set = ns_roof_ctr_new
   compute global avg_roof_disp as average of nodal displacement
end

begin user output
   node set = ns_floor_ctr_new
   compute global avg_floor_disp as average of nodal displacement
end

begin user output
   node set = ns_left_wall_ctr_new
   compute global avg_left_wall_disp as average of nodal displacement
end

begin user output
   node set = ns_right_wall_ctr_new
   compute global avg_right_wall_disp as average of nodal displacement
end

#Lines to generate the drum coordinates (x,y,z) global variable names
${ECHO(OFF)}
${num = 0}
${line1  = 'begin user output'}
```

```
${line2  = '  nodeset = {"tracker_ns_"//tostring(++num)}'}
${line3  = '  compute global {"drum_"//tostring(num)} as average '}
${line3a = 'of nodal coordinates'}
${line4  = 'end user output'}
${ECHO(ON)}
{loop(ndrums)}
  {rescan(line1)}
  {rescan(line2)}
  {rescan(line3//line3a)}
  {rescan(line4)}
{endloop}
## ~~~~~~~~~~~~~~~~~~ Results Output ~~~~~~~~~~~~~~~~~~ ##

begin results output output_1
   database name = efiles/%B-r.e
   database type = exodusII
   #Output data every 10 years of material time
   ${yr = -10}
   {Loop(101)}
   ${yr += 10}
   additional times = {(-s_1_0 + sqrt(s_1_0^2 + 2 * s_ramp * (yr * 86400.0 * 365.25))) / s_ramp}
   {EndLoop}

   global time as mom_bal_time
   global mat_time
   nodal variables = displacement as disp
   nodal variables = contact_accumulated_slip
   nodal variables = velocity
   element variables = el_avg_stress
   element variables = el_avg_mean_stress
   element variables = el_avg_eq_stress
   element variables = el_avg_eq_tr_strain
   element variables = el_avg_eq_vp_strain
   element variables = el_avg_log_strain
   element variables = el_avg_vonmises
   element variables = el_avg_eqps
   element variables = death_state
   element variables = status
   element variables = death_status
end


## ~~~~~~~~~~~~~~~~~~ History Output ~~~~~~~~~~~~~~~~~~ ##

begin history output output_3
   database name = history/%B-h.e
   database type = exodusII
   At time -0.5, increment = 0.001
   global time as mom_bal_time
   global wall_clock_time as run_time
   global mat_time
   global vp_rate_scale_factor

   global roof_disp
   global floor_disp
   global left_wall_disp
   global right_wall_disp

   global avg_roof_disp
   global avg_floor_disp
   global avg_left_wall_disp
   global avg_right_wall_disp

   global contact_energy
   global external_energy
   global internal_energy
   global kinetic_energy
end history output output_3
```

```
         begin history output output_4
            database name = history/%B_coord-h.e
            database type = exodusII
            At time -0.5, increment = 0.001
           ${ECHO(OFF)}
           #Lines to generate the history output for drum coordinates
           ${num = 0}
           ${line5 = 'global {"drum_"//tostring(++num)}'}
           ${ECHO(ON)}
           {loop(ndrums)}
             {rescan(line5)}
           {endloop}
         end history output output_4

         ## ~~~~~~~~~~~~~~~~~~~ Restart ~~~~~~~~~~~~~~~~~~~ ##

         begin restart data restart_1
            database name = restart/%B.rsout
            overwrite = True
            #This should be based on the run time specified for the job
            #I was running jobs for 48 hours so the final restart was written
            #12 minutes before the job times out (4 x 0.05hrs = 12 min)
            at wall time 0.0h increment = 11.95h
         end restart data restart_1

      end presto region SmRegion.1

   end presto procedure The_Procedure.1

end sierra WIPP Isothermal Disposal Room
```

Roof Fall Compaction Simulation

```
begin sierra WIPP Isothermal Disposal Room

   title Simulation of WIPP Isothermal Disposal Room with POP disposal canisters: Rock Fall Case
   # This model used the low disposal room horizon, rock fall geometry, and flex foam model for the
   # fiberboard along with reduction in yield strength of the #04L stainless steel and the A1008
   # drum steel. Also drum have some misalignment (1" in the x-direction) for the
   # second and third sets of drums on top of the base layer of drums

   restart = automatic

   #{KelFactor = 273.15} # convert from C to K
   #{tempK     = 300} #degK

   # Celotex tensile strength (Pa)
   #{psi_to_Pa = 6894.7573} # psi_to_Pa
   #{ten_str_psi = 150} psi
   #{tensile_strength = ten_str_psi * psi_to_Pa} Pa

   #Define the number of drums in the simulation
   ${ndrums = 153}

   #Define the momentum balance time periods
   ${t_0_0 = -5.0e-1} #s
   ${t_0_1 = 2.0} #s
   ${t_1_0 = t_0_1} #s
   ${t_1_1 = 1000.0} #s (This is an upper bound for the momentum balance time.  The simulation should terminate
   ↪  upon reaching the final material time.)

   #Define the viscoplastic scale factors
   ${s_0_0 = 2.0e-2}
   ${s_0_1 = 2.0e-2}
```

```
${s_1_0 = 2.0e-2}
${s_ramp = 2e9}

#Define the material time periods
${t_m_0_0 = t_0_0 * s_0_0}
${t_m_0_1 = s_0_0 * (t_0_1 - t_0_0) + t_m_0_0}
${t_m_1_0 = t_m_0_1} #s
${t_m_1_1 = 86400.0 * 365.25 * 1000.0} #s

${dt_trans = t_1_0}

# ============================ Directions =========================== #

define direction x with vector 1.0 0.0 0.0
define direction y with vector 0.0 1.0 0.0
define direction z with vector 0.0 0.0 1.0
define direction negative_z with vector 0.0  0.0 -1.0
define point origin with coordinates 0.0 0.0 0.0


# ============================ Functions =========================== #
begin definition for function function_constant
   type is piecewise linear
   begin values
      -1.0e16  1.0
      1.0e16  1.0
   end
end

begin definition for function fluid_density_ramp
   #Specify the density to give the appropriate lithostatic pressure at room at start of simulation and then
   ↪  ramp it down to zero
   type is analytic
   evaluate expression is "2300.0 / 2.0 * (1.0 - cos(pi * time / 0.5))"
end definition for function fluid_density_ramp


#---------- Viscoplastic Rate Scaling Functions ---------

# Sierra/SM calculates user variables at the beginning and the end of any time step where data is
# written to the exodus database.  Normally this is fine, but if variable x is an input and an
# output then x would get updated twice if data were written to the exodus database instead of
# once in a time step.  To make sure x is only calculated at the beginning (or end) of the time
# step we compare the global variable time against a user variable called stored_time.  The global
# variable time is always the time at the end of the time step.  (This is a bit weird since all
# other variables can change from the start to the end of the time step, but one can always get
# the time at the beginning of a step by calculating t-dt.)  As long as we update stored_time
# AFTER x, then the beginning of a time step corresponds to time != stored_time, and the end of
# the time step corresponds to time == stored_time.  If data is not written to the exodus database,
# then I think user variables are calculated only at the beginning of a time step.

begin function stored_time_function
   type is analytic
   expression variable: t_stored = global time
   evaluate expression = "t_stored"
end

begin function vp_rate_scale_factor_function
   #The start time column specifies when the corresponding temperature expression starts to be used.
   #(We have to add a small amount to start times to make sure the last time step
   #of the previous time period does not use the temperature expression for the next time period.)
   type is piecewise analytic
   begin expressions
      #-0.5                  "0.02"
      #1e-10                 "0.02 + 2000000000 * (time - 0)"
      {t_0_0}                "{s_0_0}"
      {t_0_1}                "{s_0_1} + {s_ramp} * (time - {dt_trans})"
   end expressions
```

```
end

begin function vp_rate_scale_factor_broadcast_function
   type is analytic
   expression variable: vp_rate_scale_factor = global vp_rate_scale_factor
   evaluate expression = "vp_rate_scale_factor"
end

begin function mat_time_function
   type is analytic
   expression variable: t = global time
   expression variable: dt = global timestep
   expression variable: t_stored = global stored_time
   expression variable: t_m = global mat_time
   expression variable: s = global vp_rate_scale_factor
   evaluate expression = "                                \#
      (t != t_stored) ? (                                    \#
      t_m + s * dt                                       \#
      ) : (                                                  \#
      t_m                                                \#
      )                                                      \#
   "
end

# ============================ Materials =========================== #

begin property specification for material halite

   density = 2300.0

   begin parameters for model md_viscoplastic

      #Intact salt parameters
      #(These Cal 3B parameters are from Table 5.2 of SAND2018-12601.)
      bulk modulus  = 20.7e9 # Pa  compute bulk modulus from E=31e+09 Pa and nu = 0.25
      shear modulus = 12.4e9  # Pa
      a0            = 5.617e1 # 1/sec
      q0/R          = 5123 # K
      n0            = 1.595
      a1            = 8.386e22 # 1/sec
      q1/R          = 12580.5 # K = 25000 cal / mol / 1.9872035 cal / (mol K)
      n1            = 5.5
      b1            = 6.086e6 # 1/sec
      a2            = 4.415e16 # 1/sec
      q2/R          = 5123 # K
      n2            = 6.279
      b2            = 3.034e-2 # 1/sec
      sigma_g       = 20570000.0 # Pa
      q             = 5335.0
      m0            = 0.9201
      k0            = 5.277e-2
      c0            = 8.882e-3 # 1/K
      m1            = 5.282
      k1            = 3.052e12
      c1            = 8.882e-3 # 1/K
      alpha_h       = 3.367
      beta_h        = -0.6838
      alpha_r       = 0.58
      a             = 16.0
      alpha         = 45.0e-6

   end parameters for model md_viscoplastic

end property specification for material halite

begin property specification for material anhydrite
   density = 2300.0 # kg / m^3
   #Sierra/SM needs to have elastic parameters defined with these names, so we repeat
   #the values in the "begin parameters ..." block.
   bulk modulus = 83.4444444e9 # Pa = bulk modulus
   shear modulus = 27814814814 # Pa = shear modulus

   #These Drucker-Prager parameters came from SAND97-0795, Table 6, except it does not
   #mention a dilation angle.  They almost certainly used the soil and foam model, which
   #has a dilatation angle of 0 degrees.  Here we have switched to associated flow.
   #The Drucker-Prager yield function is often written as phi = sqrt(J_2) + a * I_1 - C
   #B0 = bulk modulus = 83.4444444e9 Pa
   #G0 = shear modulus = 27814814814 Pa
   #C = 1.35
   #a = 0.45
   #alpha = dilatation angle = 24.227745 deg = atan(0.45)

   #Reduce Kayenta to Drucker-Prager
   begin parameters for model kayenta
      B0 = 83.4444444e9   # Pa = bulk modulus
      G0 = 27814814814    # Pa = shear modulus
      J3TYPE = 1          # Sets the dependence on J_3.  J3TYPE = 1 is a Von-Mises dependence.
      A1 = 1350000        # Pa
      A2 = 0.0
      A3 = 0.0
      A4 = 0.45
      RK = 1.0
      P0 = -1.0e99        # Put the compression cap at virtually infinity
      P1 = 0.0            # No cap
      P2 = 0.0            # No cap
      P3 = 0.0            # Zero porosity
      CR = 0.001          # Minimize the size of the curved part of the cap
      HC = 0.0            # Disable kinematic hardening
      RN = 0.0            # Disable kinematic hardening
   end parameters for model kayenta
end property specification for material anhydrite

begin definition for function polyhalite_pressure_volstrain_function
   type is piecewise linear
   ordinate is volumetric_strain
   abscissa is pressure
   begin values
      -1  -6.583333333e+10 # -65.83333333 GPa
      0  0
      1  6.583333333e+10 # 65.83333333 GPa = bulk modulus = E / (3*(1-2*nu)), where E = 55.3 GPa and nu =
      ↪  0.36
   end values
end definition for function polyhalite_pressure_volstrain_function

begin property specification for material polyhalite

   #These parameters came from SAND97-0796, Pages A-40 through A-41, Section 2.5.2

   #The yield function for this model is
   #phi = sqrt(3*J_2) - (a0 + a1 * p + a2 * p^2)
   #This model has a Drucker-Prager yield surface if a2 = 0
   #The Drucker-Prager yield function is often written as
   #phi = sqrt(J_2) + a * I_1 - C
   #Thus, we can write a0 and a1 in terms of a and C

   density = 2300.0
   begin parameters for model soil_foam
      poissons ratio  = 0.36
      youngs modulus  = 5.53e+10 # Pa
      a0              = 2459512.147 # Pa = sqrt(3) * C, where C = 1.42 MPa
      a1              = 2.457780096 # unitless = 3 * sqrt(3) * a, where a = 0.473
      a2              = 0.0
      pressure cutoff = -1000000.0 # Pa
      pressure function = polyhalite_pressure_volstrain_function
   end parameters for model soil_foam
end property specification for material polyhalite
```

```
   ##########################
   ## 304L Stainless Steel ##
   ##########################
   #
   # 18Cr-8Ni, 304L Stainless Steel Plate Per ASTM A240 (SA-240), UNS S30403
   #

   begin property specification for material SA-240_304L

      density = 7916.5 # MMPDS-01 Table 2.7.1.0(b)

      begin parameters for model ml_ep_fail
         youngs modulus = 2.02547e+11 # MMPDS-01 Table 2.7.1.0(b), Average of Tension and Compression Moduli
         ↪  and then inadvertently scaled by 1.03
         poissons ratio = 0.27 # MMPDS-01 Table 2.7.1.0(b)
         #yield stress = 2.77e8 #First point on hardening curve
         yield stress = 1.7e8
         beta = 1.0
         hardening function = SA-240_304L_Hardening_Func_20
#         critical tearing parameter = 1.64 # Consistent with last plastic strain point in the hardening curve
         critical tearing parameter = 1.6517 # for yield stress = 1.e7e8
         critical crack opening strain = 0.04828 # Engineering judgement
      end

   end

   # Hardening functions based on data obtained from ASME SA-240M Table 2(Ftu, Fty, e), INL/CON-06-11962
   # (Blandford, R. K., Morton, D. K., Snow, S. D., and Rahl, T. E., Tensile Stress-Strain Results for
   # 304L and 316L Stainless Steel Plate at Temperature, Idaho National Laboratory, 2007)


# This following hardening curve is based on a shifting of the original one to
# account for a material that has been annealed with a yield stress of 1.7e8 Pa
# which corresponds to the lowest allowable yield stress for 304L
   begin definition for function SA-240_304L_Hardening_Func_20
      type = piecewise linear
      abscissa = plastic_strain
      ordinate = yield_stress
      begin values
         0.0000E+00  1.70E+08
         1.1690E-02  2.77E+08
         1.6420E-02  3.08E+08
         2.2590E-02  3.31E+08
         3.4890E-02  3.66E+08
         5.9590E-02  4.23E+08
         1.0909E-01  5.23E+08
         1.5869E-01  6.14E+08
         2.0869E-01  6.92E+08
         2.5769E-01  7.65E+08
         3.0769E-01  8.34E+08
         3.5769E-01  9.00E+08
         4.0669E-01  9.66E+08
         4.5669E-01  1.03E+09
         5.0669E-01  1.09E+09
         1.6517E+00  2.30E+09
      end
   end

   #
   # +-+-+- ASTM A1008 Drum Steel +-+-+- #

   begin material ASTM_A1008_Drum_Steel

      density = 7750.0 # ASME B&PV Code 2015, Section II Part D, Table PRD

      begin parameters for model ml_ep_fail
```

```
         youngs modulus = 2.02000e+11 # ASME B&PV Code 2015, Section II Part D, Table TM-1 (Carbon steels with
         ↪  C <= 0.30%)
         poissons ratio = 0.300 # ASME B&PV Code 2015, Section II Part D, Table PRD
         yield stress = 1.4e8 # Corresponding to a commerical grade cold rolled steel (lowest yield stress =
         ↪  20ksi)
         beta = 1.0
         hardening function = ASTM_A1008_Drum_Steel_Hardening_Func_293
         critical tearing parameter = 1.6490 # based on yiled stress = 1.7e8Pa
         critical crack opening strain = 0.01257 # Engineering judgement
      end

   end

# Cold rolled commerical grade steel for a material with a yield stress of 1.4e8 Pa (20ksi)
   begin function ASTM_A1008_Drum_Steel_Hardening_Func_293
      type = piecewise linear
      abscissa = plastic_strain
      ordinate = yield_stress
      begin values
         0.0         1.40E+08
         0.015       1.43E+08
         6.2693E-02  2.77E+08
         6.9948E-02  2.96E+08
         7.7128E-02  3.10E+08
         8.4227E-02  3.22E+08
         8.6238E-02  3.27E+08
         1.3315E-01  3.63E+08
         1.7996E-01  3.87E+08
         2.0766E-01  4.08E+08
         2.6500E-01  4.24E+08
         1.6490E+00  8.47E+08
       end
   end
      #
   #
   # +-+-+- Plywood +-+-+- #
   #

   begin material Plywood

      # These paramters were provided by Ben Reedunn based on the paper
      #
      # "Relationships between small-specimen and large panel bending tests on structural
      # wood-based panels" by J. Dobbin McNatt and Robert W. Wellwood and Lars Bach
      # Forest Products Journal, 1990, Vol 40, Num 9, pp 10-16
      #
      # The mas stress was determined as the average of the parallel and perpendicular
      # strengths, and then it was scaled down by 20 % since wood will fail at a constant
      # load if held for long enough time, as discussed in
      #
      # "Duration of Load Revisited" by Preben Hoffmeyer and John Sorensen.
      # Wood Science and Technolology, December 2007, Vol 41, Issue 8, pp 687-711.

      density = 576.7

      begin parameters for model elastic_fracture
         youngs modulus = 5.30e+09 # Pa
         poissons ratio = 0.35 # unitless
         max stress     = 5.00e+06 # Pa
         critical crack opening strain = 0.001 # engineering judgement
      end parameters for model elastic_fracture

   end



   begin material fiberboard
```

```
      # The elastic plastic parameters were provided by Jim Bean based on the SAND report
      #
      # SAND2004-1390 by Byoung Park and Frank Hansen "Simulations of the Pipe Overpack Compute Constitutive
      ↪  Model Parameters for Use in WIPP Room Closure Calculations"
      # See Table 2
      density = 256.0 #kg/m^3

      begin parameters for model elastic_plastic
         youngs modulus = 6.895e7 # Pa
         poissons ratio = 0.05    # m/m
         yield stress = 3.447e5   # Pa
         hardening modulus = 2.413e7 # Pa
         beta = 1.0 # unitless changed from reference value of 0.5 (no idea where they got that value)
      end

      #Flex foam parameters were taken from the SNL report SAND2018-2433 by M. Neilsen, Wei-Yang Lu, Brian
      ↪  Werner, William Scherzinger, and David Lo "Flexible Foam Model" for a 15pcf polyurethane foam.
      #The paramterized Model is a "rough match" to the room temp celotex load deflection curves from the Oak
      ↪  Ridge Report Y/EN-4120 "Packaging Materials Properties Data" by M.S Walker (Jan. 1991)

      #Poissons ratio was set to zero based on a staement in a report from Westinghouse Savannah RIver company
      ↪  by A.C. Smith and P.R. Vormelker "Celotex Structural Properties Test (WSRC-TR-2000-00444)
         #"During testing, it was noted that lateral expansion(bulging) of the specimens did not occur".

      #The WSRC report discusses tensile test on dogbone and multilayer samples.  The data show a wide range of
      ↪  tensile strengths depending on factors such as relative humidity, temperature and
      #whether loading is parallel or perpindicular to the celotex sheets. When multiple sheets are glued
      ↪  together the glue strength plays a large role (reduced strength) when loading is perpindicular to the
      ↪  sheets.

      begin parameters for model flex_foam
         youngs modulus           = {120000.0* psi_to_Pa}  #Pa
         poissons ratio           = 0.0000 #0.250
         phi                      = 0.200
         flow rate                = 1.000
         power exponent           = 1.000
         dev multiplier           = 0.2
         tensile strength         = {tensile_strength} #Pa
         adam                     = 1.0
         bdam                     = 0.5
         youngs function          = foam_yModulus
         poissons function        = foam_Constant
         youngs phi function      = foam_E
         poissons phi function    = foam_Constant
         rate function            = foam_Rate
         exponent function        = foam_Expo
         shear hardening function = foam_Shear
         hydro hardening function = foam_Hydro
         beta function            = foam_Beta
         dmod function            = foam_Modulus
         dpr function             = foam_Constant
         dmod phi function        = foam_E
         dpr phi function         = foam_Constant
         damage function          = foam_Damage
      end parameters for model flex_foam
end
##
## currently no damage = no failure of foam in tension
##

begin definition for function foam_Damage
  type is piecewise linear
  begin values
     0.00000     0.00000
     0.14000     0.00000
     0.40000     0.00000
   100.00000     0.00000
  end values
end definition for function foam_Damage

begin definition for function foam_Beta
  type is piecewise linear
   begin values
      0.000        0.600
      0.200        0.600
      0.225        0.600
      0.250        0.595
      0.275        0.590
      0.300        0.575
      0.325        0.540
      0.350        0.460
      0.375        0.380
      0.400        0.300
      0.425        0.220
      0.450        0.140
      0.475        0.100
      1.000        0.100
     10.000        0.100
   end values
end definition for function foam_Beta

begin definition for function foam_E
   type is piecewise linear
    begin values
      0.00        0.020
      0.20        0.020
      0.25        0.018
      0.30        0.016
      0.35        0.028
      0.40        0.078
      0.45        0.148
      0.50        0.238
      0.55        0.348
      0.60        0.478
      0.65        0.628
      0.70        0.798
      0.80        1.050
      0.90        1.400
      1.00        1.780
      1.50        3.000
      2.00        6.000
     10.00       10.000
    end values
end

##
## currently just using 60 ppm/C based on measured rigid urethane foam
## flexible should probably be higher.
##

begin definition for function foam_Thermal
     type is piecewise linear
     ordinate is strain
     abscissa is temperature
     begin values
      {-500.0 + KelFactor}  -0.0300
      {   0.0 + KelFactor}   0.0000
      { 500.0 + KelFactor}   0.0300
     end values
end definition for function foam_Thermal

begin definition for function foam_yModulus
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
```

```
      {-53.90 + KelFactor}      0.060
      {-40.00 + KelFactor}      0.050
      {-20.00 + KelFactor}      0.030
      {  0.00 + KelFactor}      0.021
      { 21.10 + KelFactor}      0.018
      { 73.90 + KelFactor}      0.015
    end values
end definition for function foam_yModulus


begin definition for function foam_Modulus
  type is piecewise linear
   begin values
       {-53.90 + KelFactor}       6.000
       {-40.00 + KelFactor}       1.000
       {-20.00 + KelFactor}       0.080
       {  0.00 + KelFactor}       0.048
       { 21.10 + KelFactor}       0.036
       { 73.90 + KelFactor}       0.033
   end values
end definition for function foam_Modulus

begin definition for function foam_Constant
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
       {-53.90 + KelFactor}      1.0
       { 21.10 + KelFactor}      1.0
       { 73.90 + KelFactor}      1.0
    end values
end definition for function foam_Constant

begin definition for function foam_Rate
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
       {-53.90 + KelFactor}     -15.00
       { 21.10 + KelFactor}      4.80
       { 73.90 + KelFactor}      8.80
    end values
end definition for function foam_Rate

begin definition for function foam_Expo
    type is piecewise linear
    ordinate is temperature
    abscissa is time
    begin values
       {-53.90 + KelFactor}      8.5
       { 21.10 + KelFactor}      5.0
       { 73.90 + KelFactor}      3.0
    end values
end definition for function foam_Expo

begin definition for function foam_Shear
    type is piecewise linear
    begin values
       0.000      {120.0 * psi_to_Pa}
       0.400      {120.0 * psi_to_Pa}
       0.450      {130.0 * psi_to_Pa}
       0.500      {180.0 * psi_to_Pa}
       0.550      {230.0 * psi_to_Pa}
       1.000      {800.0 * psi_to_Pa}
      10.000     {1000.0 * psi_to_Pa}
    end values
end definition for function foam_Shear
```

```
begin definition for function foam_Hydro
    type is piecewise linear
    begin values
       0.000      {100.0 * psi_to_Pa}
       0.400      {100.0 * psi_to_Pa}
       0.450      {130.0 * psi_to_Pa}
       0.500      {200.0 * psi_to_Pa}
       0.550      {250.0 * psi_to_Pa}
       1.000     {1000.0 * psi_to_Pa}
      10.000     {2000.0 * psi_to_Pa}
    end values
end definition for function foam_Hydro


   ## ~~~~~~~~~~~~~~~~~~ Section Definitions ~~~~~~~~~~~~~~~~~~ ##

   begin solid section solid_1
      formulation = selective_deviatoric
      #formulation = mean_quadrature
      strain incrementation = strongly_objective
   end solid section solid_1

   #wall of pipe
   begin shell section shell_t6p35mm #
      thickness = 0.00635# 0.25 in.
      lofting factor = 0.5
   end

   #bottom of pipe
   begin shell section shell_t7p112mm #
      thickness = 0.007112# 0.28 in.
      lofting factor = 0.5
   end

   #drum
   begin shell section shell_t1p5mm
      thickness = 0.0015
      lofting factor = 0.5
   end


   ## ~~~~~~~~~~~~~~~~~~ Finite Element Model ~~~~~~~~~~~~~~~~~~ ##

   begin finite element model room

      database name = mesh/disposal_room_pop_assembly_lowerHorizon_rockFall.g
      database type = exodusII

      begin block defaults
         effective moduli model = elastic
      end

      #
      begin parameters for block b_halite
         material halite
         solid mechanics use model md_viscoplastic
         section = solid_1
      end
      #
      begin parameters for block b_arg_halite
         #material arg_halite
         material halite
         solid mechanics use model md_viscoplastic
         section = solid_1
      end

      begin parameters for block RockFall_upper_tri
         material halite
```

```
      solid mechanics use model md_viscoplastic
      section = solid_1
   end

   begin parameters for block RockFall_lower
      material halite
      solid mechanics use model md_viscoplastic
      section = solid_1
   end

   begin parameters for block b_anhydrite
      material anhydrite
      solid mechanics use model kayenta
      section = solid_1
   end
   #
   begin parameters for block b_polyhalite
      material polyhalite
      solid mechanics use model soil_foam
      section = solid_1
   end
   #
   begin parameters for block \#
      pop_drum_wall   pop_drum_base   pop_drum_lid \#
      pop_drum_wall_2 pop_drum_base_2 pop_drum_lid_2
      material ASTM_A1008_Drum_Steel
      solid mechanics use model ml_ep_fail
      section = shell_t1p5mm
   end

   begin parameters for block pop_cv_top_flange pop_cv_top_flange_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      section = solid_1
   end

   begin parameters for block pop_cv_wall pop_cv_wall_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      section = shell_t6p35mm
   end

   begin parameters for block pop_cv_base pop_cv_base_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      section = shell_t7p112mm
   end

   begin parameters for block \#
      pop_plywood_lower   pop_plywood_upper \#
      pop_plywood_lower_2 pop_plywood_upper_2
      material Plywood
      solid mechanics use model elastic_fracture
      section = solid_1
   end

   begin parameters for block \#
      pop_fiberboard_lower   pop_fiberboard_upper   pop_fiberboard_side   pop_fiberboard_flange \#
      pop_fiberboard_lower_2 pop_fiberboard_upper_2 pop_fiberboard_side_2 pop_fiberboard_flange_2
      material fiberboard
      #solid mechanics use model elastic_plastic
      solid mechanics use model flex_foam
      section = solid_1
   end

end
```

```
## ~~~~~~~~~~~~~~~~~~ Time Step Control ~~~~~~~~~~~~~~~~~~ ##

begin presto procedure the_procedure.1

   begin time control

      begin time stepping block p0
         start time = -0.5
         begin parameters for presto region SmRegion.1
            time step scale factor = 1
            time step increase factor = 5
         end
      end

      begin time stepping block p1
         start time = 0
         begin parameters for presto region SmRegion.1
            time step scale factor = 1
            time step increase factor = 5
         end
      end

      termination time = 1000

   end


   ## ~~~~~~~~~~~~~~~~~~ Begin Analysis ~~~~~~~~~~~~~~~~~~ ##

   begin presto region SmRegion.1

      begin solution termination
         terminate global mat_time > 3.15576e+10
      end

      use finite element model room

      begin node based time step parameters nbased_tstep
      end


      ## ~~~~~~~~~~~~~~~~~~ Analytic Objects ~~~~~~~~~~~~~~~~~~ ##

      begin analytic object pos_y_sym_analytic_plane
         type = plane
         normal direction = 0.0 -1.0 0.0
         origin = 0.0 1.07886 0.0
         scale factor = 0.5
      end

      begin analytic object neg_y_sym_analytic_plane
         type = plane
         normal direction = 0.0 1.0 0.0
         origin = 0.0 -1.07886 0.0
         scale factor = 0.5
      end


      ## ~~~~~~~~~~~~~~~~~~ Boundary Conditions ~~~~~~~~~~~~~~~~~~ ##

      begin gravity
         include all blocks
         gravitational constant = 9.79  # SAND88-2948 used 9.79 m/s^2.  9.79 m/s^2 is close to the local
         ↪  value of gravity at WIPP (9.79483 m/s^2).
         direction = negative_z
         function = function_constant
      end
```

```
begin prescribed temperature
   include all blocks
   function = function_constant
   scale factor = 300.0
end

begin pressure
   surface = ss_top_surf
   function = function_constant
   scale factor = 13.57e+06
end

begin fixed displacement
   node set = ns_bot_surf
   components = z
end

begin fixed displacement
   node set = ns_left_surf
   components = x
end

begin fixed displacement
   node set = ns_right_surf
   components = x
end

begin fixed displacement
   node set = ns_back_surf
   components = y
end

begin fixed displacement
   node set = ns_front_surf
   components = y
end

begin fluid_pressure
   surface = ss_room_surf
   active periods = p0
   #Specify which direction corresponds to the depth of fluid.
   fluid surface normal = z
   #The depth function is defined relative to the reference point.
   #See the figure in the fluid pressure section of the Sierra/SM manual for more info.
   reference point = origin
   #Set the fluid height, relative to the reference point, to give the appropriate lithostatic
   ↪  pressure at room
   #p1 = -15.97e6 Pa
   #p2 = -13.57e6 Pa
   #z1 = -54.19 m
   #z2 = 52.87 m
   #slope = (p2-p1)/(z2-z1)
   #p = slope*(z-z1)+p1
   #z0 = depth = elevation of fluid surface
   #depth = -p1 / slope + z1
   depth = 658.2050833333333 #m = height
   #Specify the function to ramp the fluid density down by time = 0
   density function = fluid_density_ramp
   gravitational constant = 9.79
end


## ~~~~~~~~~~~~~~~~~~ CONTACT MDOEL ~~~~~~~~~~~~~~~~~~ ##

begin contact definition contact_defn

   enforcement = al
```

```
begin enforcement options
   momentum balance iterations = 20
end

#---------- Contact Surface Definitions ----------

begin contact surface pop_surf
   block = pop_drum_wall \#
           pop_drum_base \#
           pop_drum_lid \#
           pop_plywood_lower \#
           pop_plywood_upper \#
           pop_cv_wall \#
           pop_cv_base \#
           pop_cv_top_flange \#
           pop_fiberboard_lower \#
           pop_fiberboard_upper \#
           pop_fiberboard_side \#
           pop_fiberboard_flange
end

begin contact surface pop_surf_2
   block = pop_drum_wall_2 \#
           pop_drum_base_2 \#
           pop_drum_lid_2 \#
           pop_plywood_lower_2 \#
           pop_plywood_upper_2 \#
           pop_cv_wall_2 \#
           pop_cv_base_2 \#
           pop_cv_top_flange_2 \#
           pop_fiberboard_lower_2 \#
           pop_fiberboard_upper_2 \#
           pop_fiberboard_side_2 \#
           pop_fiberboard_flange_2
end


begin contact surface room_surf
   surface = ss_room_surf
end

contact surface clay_A_master contains SS_clay_A_master
contact surface clay_A_slave contains SS_clay_A_slave

contact surface clay_B_master contains SS_clay_B_master
contact surface clay_B_slave contains SS_clay_B_slave

contact surface clay_D_master contains SS_clay_D_master
contact surface clay_D_slave contains SS_clay_D_slave

contact surface clay_E_master contains SS_clay_E_master
contact surface clay_E_slave contains SS_clay_E_slave

contact surface clay_F_master contains SS_clay_F_master
contact surface clay_F_slave contains SS_clay_F_slave

contact surface clay_G_master contains SS_clay_G_master
contact surface clay_G_slave contains SS_clay_G_slave

contact surface clay_H_master contains SS_clay_H_master
contact surface clay_H_slave contains SS_clay_H_slave

contact surface clay_I_master contains SS_clay_I_master
contact surface clay_I_slave contains SS_clay_I_slave

contact surface clay_J_master contains SS_clay_J_master
contact surface clay_J_slave contains SS_clay_J_slave
```

```
            contact surface clay_K_master contains SS_clay_K_master
            contact surface clay_K_slave contains SS_clay_K_slave

            contact surface clay_L_master contains SS_clay_L_master
            contact surface clay_L_slave contains SS_clay_L_slave

            contact surface clay_M_master contains SS_clay_M_master
            contact surface clay_M_slave contains SS_clay_M_slave

# Note that these regions directly above the disposal room are actually
# arg.halite but it doesnt matter because the name isn't important and
# the material parameters are the same for pure halite and arg. halite
            contact surface Halite_1_master contains SS_Halite_M1
            contact surface Halite_1_slave  contains SS_Halite_S1

            contact surface Halite_2_master contains SS_Halite_M2
            contact surface Halite_2_slave  contains SS_Halite_S2

            contact surface pos_y_sym_contact_surface contains pos_y_sym_analytic_plane

            contact surface neg_y_sym_contact_surface contains neg_y_sym_analytic_plane


            #---------- Friction Model Definitions ----------

            begin constant friction model clay_seam_model
               friction coefficient = 0.20
               #friction coefficient = 0.00
            end

            begin constant friction model salt_to_salt_friction_model
               friction coefficient = 0.50
            end

            begin constant friction model metal_to_salt_friction_model
               friction coefficient = 0.4 #0.65
            end

            begin constant friction model metal_to_metal_friction_model
               friction coefficient = 0.30 #0.50
            end

            begin tied model tied_contact
            end tied model tied_contact

            BEGIN FRICTIONLESS MODEL no_friction
            END FRICTIONLESS MODEL no_friction

            begin time variant model time_varying_friction
              model = tied_contact during periods p0
              #model = no_friction during periods p1
              model = salt_to_salt_friction_model during periods p1
            end time variant model time_varying_friction

            begin remove initial overlap
               #          debug iteration plot = ON
               overlap iterations = 10
               overlap normal tolerance     = 1.0e-3
               overlap tangential tolerance = 1.0e-3
            end remove initial overlap


            #---------- Contact Interaction Definitions ----------

            begin interaction defaults
               constraint formulation = face_face
               general contact = on
            end
```

```
#JEB adding new lines for rockfall case
begin interaction Rockfall_1
  master = Halite_1_master
  slave  = Halite_1_slave
  friction model = tied_contact
  # add some tolerance to see if I can get the surfaces
  # to be tied
  #normal tolerance  = 1.0e-3
  #capture tolerance = 1.0e-3
end interaction Rockfall_1

begin interaction Rockfall_2
  master = Halite_2_master
  slave  = Halite_2_slave
  friction model = time_varying_friction
end interaction Rockfall_2

begin interaction clay_A
   master = clay_A_master
   slave = clay_A_slave
   friction model = clay_seam_model
end interaction

begin interaction clay_B
   master = clay_B_master
   slave = clay_B_slave
   friction model = clay_seam_model
end interaction

begin interaction clay_D
   master = clay_D_master
   slave = clay_D_slave
   friction model = clay_seam_model
end

begin interaction clay_E
   master = clay_E_master
   slave = clay_E_slave
   friction model = clay_seam_model
end

begin interaction clay_F
   master = clay_F_master
   slave = clay_F_slave
   friction model = clay_seam_model
end

begin interaction clay_G
   master = clay_G_master
   slave = clay_G_slave
   friction model = clay_seam_model
end

begin interaction clay_H
   master = clay_H_master
   slave = clay_H_slave
   friction model = clay_seam_model
end

begin interaction clay_I
   master = clay_I_master
   slave = clay_I_slave
   friction model = clay_seam_model
end

begin interaction clay_J
   master = clay_J_master
```

```
      slave = clay_J_slave
      friction model = clay_seam_model
   end

   begin interaction clay_K
      master = clay_K_master
      slave = clay_K_slave
      friction model = clay_seam_model
   end

   begin interaction clay_L
      master = clay_L_master
      slave = clay_L_slave
      friction model = clay_seam_model
   end

   begin interaction clay_M
      master = clay_M_master
      slave = clay_M_slave
      friction model = clay_seam_model
   end interaction

   begin interaction salt_to_salt
      surfaces = room_surf room_surf clay_G_master clay_G_slave clay_F_master clay_F_slave
      friction model = salt_to_salt_friction_model
   end

   begin interaction metal_to_salt
      master = room_surf
      slave  = pop_surf
      friction model = metal_to_salt_friction_model
   end

   begin interaction metal_to_salt_2
      master = room_surf
      slave  = pop_surf_2
      friction model = metal_to_salt_friction_model
   end

   begin interaction metal_to_metal
      # self contact
      surfaces = pop_surf pop_surf
      friction model = metal_to_metal_friction_model
   end

   begin interaction metal_to_metal_2
      # self contact
      surfaces = pop_surf_2 pop_surf_2
      friction model = metal_to_metal_friction_model
   end
   begin interaction metal_to_metal_3
      # contact between 2 groups of canister blocks
      surfaces = pop_surf pop_surf_2
      friction model = metal_to_metal_friction_model
   end
   begin interaction metal_to_analytic_plane
      # contact between full drum blocks and plane
      surfaces = pop_surf pop_surf_2 pos_y_sym_contact_surface neg_y_sym_contact_surface
      friction model = metal_to_metal_friction_model
   end

end


## ~~~~~~~~~~~~~~~~~~ Initial Conditions ~~~~~~~~~~~~~~~~~~ ##

begin initial condition
   block = b_halite b_arg_halite b_polyhalite b_anhydrite RockFall_upper_tri RockFall_lower
```

```
            #It is important to initialize unrotated_stress and not stress, because
            #stress is just an output that gets calculated as needed, while
            #unrotated_stress gets stored and used by the constitutive models.
            #As of November 2015, initializing stress does nothing.
            initialize variable name = unrotated_stress
            variable type = element
            subroutine real parameter: top = 52.87
            subroutine real parameter: bot = -54.19
            subroutine real parameter: p1 = -13.57e6
            subroutine real parameter: po = -15.97e6
            subroutine real parameter: kvert_xx = 1.0
            subroutine real parameter: kvert_yy = 1.0
            subroutine real parameter: kvert_zz = 1.0
            subroutine real parameter: kvert_xy = 0.0
            subroutine real parameter: kvert_yz = 0.0
            subroutine real parameter: kvert_zx = 0.0
            subroutine string parameter: dir = Z
            element block subroutine = geo_is
         end initial condition

         #--------- Viscoplastic Rate Scaling ---------

         # See note above stored_time_function.
         begin user variable stored_time
            type = global real length = 1
            global operator = min
            initial value = -0.5
         end

         begin user variable mat_time
            type = global real length = 1
            global operator = min
            #edit the following line for restart runs!
            initial value = -0.01
         end

         begin user variable vp_rate_scale_factor
            type = global real length = 1
            global operator = min
            initial value = 0.02
         end


         ## ~~~~~~~~~~~~~~~~~~ Element Death ~~~~~~~~~~~~~~~~~~ ##
         # Use material criterion to kill wood elements with elastic fracture model
         begin element death pop_wood_failure
            block = pop_plywood_lower   pop_plywood_upper \#
                    pop_plywood_lower_2 pop_plywood_upper_2
            material criterion = on kill when 1 integration points remain
         end

         begin element death pop_drum_failure
            block = pop_drum_wall   pop_drum_base   pop_drum_lid \#
                    pop_drum_wall_2 pop_drum_base_2 pop_drum_lid_2
            material criterion = on kill when 1 integration points remain
         end

         begin element death pop_cv_failure
            block = pop_cv_wall   pop_cv_base   pop_cv_top_flange    \#
                    pop_cv_wall_2 pop_cv_base_2 pop_cv_top_flange_2
            material criterion = on kill when 1 integration points remain
         end

#JEB uncomment when a failure model for the fiberboards is included
#          begin element death pop_fiberboard
#             block = pop_fiberboard_upper   pop_fiberboard_lower   pop_fiberboard_side   pop_fiberboard_flange
↪  \#
```

```
#                      pop_fiberboard_upper_2 pop_fiberboard_lower_2 pop_fiberboard_side_2
↪  pop_fiberboard_flange_2
#          end

         begin element death large_equivalent_stress
             block = b_halite b_arg_halite RockFall_upper_tri RockFall_lower
             criterion is element value of el_avg_eq_stress > 50.0e6
         end element death large_equivalent_stress

         begin element death degenerate_element
            include all blocks
            death on inversion = on
            criterion is element value of element_shape <= 0.0
         end


         ## ~~~~~~~~~~~~~~~~~~~ User Output Variable Definitions ~~~~~~~~~~~~~~~~~~~ ##

         begin user output
            block = b_halite b_arg_halite RockFall_upper_tri RockFall_lower
            compute global vp_rate_scale_factor as function vp_rate_scale_factor_function
            # Although vp_scale_factor is really a global variable, it needs to be at the integration points
            # so the material model can read it.
            $0
            compute element vp_rate_scale_factor(1) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(2) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(3) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(4) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(5) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(6) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(7) as function vp_rate_scale_factor_broadcast_function
            compute element vp_rate_scale_factor(8) as function vp_rate_scale_factor_broadcast_function
            compute at every step
         end

         begin user output
            compute global mat_time as function mat_time_function
            # See note above stored_time_function.  Note that stored_time must be calculated
            # AFTER other functions that use it.
            compute global stored_time as function stored_time_function
            compute at every step
         end


         begin user output
            compute element el_avg_stress as average of element cauchy_stress
            compute element el_avg_mean_stress as average of element hydrostatic_stress
            compute element el_avg_eq_stress as average of element eq_stress
            compute element el_avg_eq_tr_strain as average of element eq_tr_strain
            compute element el_avg_eq_vp_strain as average of element eq_vp_strain
            compute element el_avg_log_strain as average of element unrotated_log_strain
            compute element el_avg_vonmises as average of element von_mises
            compute element el_avg_eqps as average of element eqps
         end

         begin user output
            node set = ns_roof_ctr
            compute global roof_disp as average of nodal displacement
         end

         begin user output
            node set = ns_floor_ctr
            compute global floor_disp as average of nodal displacement
         end

         begin user output
            node set = ns_left_wall_ctr
            compute global left_wall_disp as average of nodal displacement
```

```
end

begin user output
   node set = ns_right_wall_ctr
   compute global right_wall_disp as average of nodal displacement
end

#These 4 user outputs are for computing average displacements along line of nodes
begin user output
   node set = ns_roof_ctr_new
   compute global avg_roof_disp as average of nodal displacement
end

begin user output
   node set = ns_floor_ctr_new
   compute global avg_floor_disp as average of nodal displacement
end

begin user output
   node set = ns_left_wall_ctr_new
   compute global avg_left_wall_disp as average of nodal displacement
end

begin user output
   node set = ns_right_wall_ctr_new
   compute global avg_right_wall_disp as average of nodal displacement
end

#Lines to generate the drum coordinates (x,y,z) global variable names
${ECHO(OFF)}
${num = 0}
${line1  = 'begin user output'}
${line2  = '  nodeset = {"tracker_ns_"//tostring(++num)}'}
${line3  = '  compute global {"drum_"//tostring(num)} as average '}
${line3a = 'of nodal coordinates'}
${line4  = 'end user output'}
${ECHO(ON)}
{loop(ndrums)}
  {rescan(line1)}
  {rescan(line2)}
  {rescan(line3//line3a)}
  {rescan(line4)}
{endloop}
## ~~~~~~~~~~~~~~~~~~~ Results Output ~~~~~~~~~~~~~~~~~~~ ##

begin results output output_1
   database name = efiles/%B-r.e
   database type = exodusII
   $a = {a = s_ramp/2}
   $b = {b = s_1_0 - s_ramp * t_1_0}
   $secperyr = {secperyr = 86400.0 * 365.25}
   # output some early time to get an idea of the
   # room roof rock fall (all of these times are in momemtum balance time (sec)

   #Save some of the steps prior to using larger viscoplastic scaling
   #Save from 0 to 2 seconds of momentum balance time with an interval of 0.05s
   ${sec = -0.05}
   {Loop(41)}
   additional times = {sec += 0.05} # (sec)
   {EndLoop}

   ${yr = 0}
   #Output data every 1 years of material time (time output is momemtum balance time (sec))
   {Loop(1000)}
   ${yr += 1} # yr of material time
   additional times = {(-b + sqrt(b*b - 4 * a * (-yr * secperyr + t_m_1_0 - (s_1_0 * t_1_0 - 0.5 *
   ↪  s_ramp * t_1_0 * t_1_0)) ))/(2*a)} # seconds of momentum balance time
   {EndLoop}
```

```
   global time as mom_bal_time
   global mat_time
   nodal variables = displacement as disp
   nodal variables = contact_accumulated_slip
   nodal variables = velocity
   element variables = el_avg_stress
   element variables = el_avg_mean_stress
   element variables = el_avg_eq_stress
   element variables = el_avg_eq_tr_strain
   element variables = el_avg_eq_vp_strain
   element variables = el_avg_log_strain
   element variables = el_avg_vonmises
   element variables = el_avg_eqps
   element variables = death_state
   element variables = status
   element variables = death_status
end


## ~~~~~~~~~~~~~~~~~~ History Output ~~~~~~~~~~~~~~~~~~ ##

begin history output output_3
   database name = history/%B-h.e
   database type = exodusII
   At time -0.5, increment = 0.001
   global time as mom_bal_time
   global wall_clock_time as run_time
   global mat_time
   global vp_rate_scale_factor

   global roof_disp
   global floor_disp
   global left_wall_disp
   global right_wall_disp

   global avg_roof_disp
   global avg_floor_disp
   global avg_left_wall_disp
   global avg_right_wall_disp

   global contact_energy
   global external_energy
   global internal_energy
   global kinetic_energy
end history output output_3

begin history output output_4
   database name = history/%B_coord-h.e
   database type = exodusII
   global mat_time
   global time as mom_bal_time
   At time -0.5, increment = 0.001
  ${ECHO(OFF)}
  #Lines to generate the history output for drum coordinates
  ${num = 0}
  ${line5 = 'global {"drum_"//tostring(++num)}'}
  ${ECHO(ON)}
  {loop(ndrums)}
    {rescan(line5)}
  {endloop}
end history output output_4

## ~~~~~~~~~~~~~~~~~~ Restart ~~~~~~~~~~~~~~~~~~ ##

begin restart data restart_1
   database name = restart/%B.rsout
```

```
            overwrite = True
            #This should be based on the run time specified for the job
            #I was running jobs for 48 hours so the final restart was written
            #12 minutes before the job times out (4 x 0.05hrs = 12 min)
            at wall time 0.0h increment = 11.95h
         end restart data restart_1

      end presto region SmRegion.1

   end presto procedure The_Procedure.1

end sierra WIPP Isothermal Disposal Room
```

APPENDIX B. Pipe Centroid Coordinates

Tables B-1 and B-2 list the pipe centroid coordinates at *t* = 0 and 1000 yr for the respective 6-inch and 12-inch Standard container gradual compaction simulations. See Section 3.2 for a description of how the pipe centroids were computed.

Table B-1. 6-inch Standard container pipe centroid coordinates

	t = 0 yr			*t* = 1000 yr		
Pipe #	*X* (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
1	-2.060	-0.000	-5.170	-2.320	-0.179	-5.390
2	-1.480	-0.000	-5.170	-1.930	-0.129	-5.430
3	-1.770	0.502	-5.170	-1.770	0.213	-5.470
4	-2.350	0.502	-5.170	-2.290	0.350	-5.510
5	-2.640	0.000	-5.170	-2.450	-0.254	-5.380
6	-2.350	-0.502	-5.170	-2.130	-0.866	-5.380
7	-1.770	-0.502	-5.170	-1.900	-0.868	-5.440
8	0.686	-0.000	-5.170	0.805	-0.280	-5.570
9	1.270	-0.000	-5.170	1.380	0.342	-5.540
10	0.975	0.502	-5.170	1.250	0.638	-5.590
11	0.396	0.502	-5.170	0.766	0.819	-5.580
12	0.106	-0.000	-5.170	-0.198	-0.064	-5.580
13	0.396	-0.502	-5.170	0.782	-0.667	-5.580
14	0.975	-0.502	-5.170	0.840	-0.944	-5.540
15	3.430	-0.000	-5.170	2.550	0.116	-5.370
16	4.010	-0.000	-5.170	2.740	-0.170	-5.260
17	3.720	0.502	-5.170	2.790	0.889	-5.280
18	3.140	0.502	-5.170	2.680	0.753	-5.400
19	2.850	0.000	-5.170	2.320	0.105	-5.420
20	3.140	-0.502	-5.170	2.600	-0.901	-5.360
21	3.720	-0.502	-5.170	2.750	-0.941	-5.280
22	-2.060	-0.000	-5.990	-1.920	-0.234	-5.540
23	-1.480	-0.000	-5.990	-1.400	-0.108	-5.540
24	-1.770	0.502	-5.990	-1.770	0.576	-5.510
25	-2.350	0.502	-5.990	-1.940	0.392	-5.560
26	-2.640	0.000	-5.990	-2.320	0.002	-5.500
27	-2.350	-0.502	-5.990	-2.230	-0.549	-5.410
28	-1.770	-0.502	-5.990	-1.800	-0.518	-5.540
29	0.686	-0.000	-5.990	0.715	-0.004	-5.590
30	1.270	-0.000	-5.990	1.230	-0.188	-5.540
31	0.975	0.502	-5.990	0.958	0.532	-5.570
32	0.396	0.502	-5.990	0.385	0.520	-5.580
33	0.106	0.000	-5.990	0.203	-0.036	-5.640
34	0.396	-0.502	-5.990	0.351	-0.750	-5.580

Table B-1. 6-inch Standard container pipe centroid coordinates

	t = 0 yr			*t* = 1000 yr		
Pipe #	*X* (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
35	0.975	-0.502	-5.990	1.050	-0.560	-5.560
36	3.430	-0.000	-5.990	2.670	-0.105	-5.510
37	4.010	-0.000	-5.990	2.970	0.131	-5.570
38	3.720	0.502	-5.990	2.860	0.553	-5.480
39	3.140	0.502	-5.990	2.390	0.531	-5.520
40	2.850	0.000	-5.990	2.020	0.097	-5.470
41	3.140	-0.502	-5.990	2.500	-0.514	-5.450
42	3.720	-0.502	-5.990	2.770	-0.574	-5.420
43	-2.060	-0.000	-4.340	-2.100	0.015	-5.450
44	-1.480	-0.000	-4.340	-1.570	0.055	-5.540
45	-1.770	0.502	-4.340	-1.950	0.637	-5.480
46	-2.350	0.502	-4.340	-2.140	0.576	-5.380
47	-2.640	0.000	-4.340	-2.430	0.148	-5.290
48	-2.350	-0.502	-4.340	-2.300	-0.461	-5.400
49	-1.770	-0.502	-4.340	-1.930	-0.413	-5.480
50	0.686	-0.000	-4.340	1.100	0.180	-5.590
51	1.270	-0.000	-4.340	1.550	-0.136	-5.540
52	0.975	0.502	-4.340	1.130	0.684	-5.550
53	0.396	0.502	-4.340	0.795	0.453	-5.620
54	0.106	0.000	-4.340	0.242	-0.221	-5.600
55	0.396	-0.502	-4.340	0.276	-0.525	-5.620
56	0.975	-0.502	-4.340	1.350	-0.509	-5.530
57	3.430	-0.000	-4.340	2.790	-0.229	-5.030
58	4.010	-0.000	-4.340	2.970	0.127	-4.850
59	3.720	0.502	-4.340	2.890	0.484	-4.960
60	3.140	0.502	-4.340	2.640	0.347	-5.290
61	2.850	0.000	-4.340	2.490	-0.251	-5.250
62	3.140	-0.502	-4.340	2.670	-0.492	-5.220
63	3.720	-0.502	-4.340	2.820	-0.558	-5.010
64	-3.430	-0.792	-5.170	-2.540	-0.931	-5.380
65	-2.850	-0.792	-5.170	-2.320	-0.990	-5.330
66	-3.140	-0.290	-5.170	-2.470	-0.472	-5.360
67	-3.720	-0.290	-5.170	-2.650	-0.452	-5.330
68	-4.010	-0.792	-5.170	-2.780	-0.708	-5.240
69	-0.686	-0.792	-5.170	-1.280	-0.783	-5.580
70	-0.106	-0.792	-5.170	-0.108	-0.523	-5.600
71	-0.396	-0.290	-5.170	-0.780	-0.318	-5.570
72	-0.975	-0.290	-5.170	-1.490	-0.440	-5.560
73	-1.270	-0.792	-5.170	-1.620	-0.709	-5.530
74	2.060	-0.792	-5.170	2.290	-0.600	-5.530
75	2.640	-0.792	-5.170	2.500	-0.988	-5.380
76	2.350	-0.290	-5.170	2.160	-0.115	-5.500
77	1.770	-0.290	-5.170	1.730	-0.202	-5.460
78	1.480	-0.792	-5.170	1.150	-0.970	-5.520
79	-3.430	0.792	-5.170	-2.430	0.926	-5.360
80	-2.850	0.792	-5.170	-1.920	0.922	-5.370
81	-4.010	0.792	-5.170	-2.750	0.866	-5.270
82	-3.720	0.290	-5.170	-2.690	0.496	-5.320

Table B-1. 6-inch Standard container pipe centroid coordinates

	t = 0 yr			t = 1000 yr		
Pipe #	X (m)	Y (m)	Z (m)	X (m)	Y (m)	Z (m)
83	-3.140	0.290	-5.170	-2.390	0.140	-5.460
84	-0.686	0.792	-5.170	-1.150	0.951	-5.540
85	-0.106	0.792	-5.170	-0.216	0.601	-5.570
86	-1.270	0.792	-5.170	-1.650	0.838	-5.490
87	-0.975	0.290	-5.170	-1.470	0.480	-5.530
88	-0.396	0.290	-5.170	-0.794	0.333	-5.570
89	2.060	0.792	-5.170	2.220	0.990	-5.340
90	2.640	0.792	-5.170	2.530	0.997	-5.400
91	1.480	0.792	-5.170	1.090	0.905	-5.520
92	1.770	0.290	-5.170	1.850	0.592	-5.400
93	2.350	0.290	-5.170	2.180	0.466	-5.460
94	-3.430	-0.792	-5.990	-2.710	-0.681	-5.550
95	-2.850	-0.792	-5.990	-2.130	-0.702	-5.560
96	-3.140	-0.290	-5.990	-2.630	-0.321	-5.530
97	-3.720	-0.290	-5.990	-2.980	-0.294	-5.550
98	-4.010	-0.792	-5.990	-3.020	-0.753	-5.590
99	-0.686	-0.792	-5.990	-0.537	-0.673	-5.560
100	-0.106	-0.792	-5.990	-0.262	-0.950	-5.580
101	-0.396	-0.290	-5.990	-0.402	-0.267	-5.640
102	-0.975	-0.290	-5.990	-1.050	-0.140	-5.580
103	-1.270	-0.792	-5.990	-1.440	-0.981	-5.520
104	2.060	-0.792	-5.990	1.980	-0.917	-5.480
105	2.640	-0.792	-5.990	2.210	-0.957	-5.520
106	2.350	-0.290	-5.990	2.080	-0.316	-5.410
107	1.770	-0.290	-5.990	1.680	-0.535	-5.500
108	1.480	-0.792	-5.990	1.490	-0.764	-5.510
109	-3.430	0.792	-5.990	-2.640	0.707	-5.530
110	-2.850	0.792	-5.990	-2.090	0.772	-5.510
111	-4.010	0.792	-5.990	-2.970	0.755	-5.550
112	-3.720	0.290	-5.990	-2.860	0.134	-5.470
113	-3.140	0.290	-5.990	-2.400	0.540	-5.480
114	-0.686	0.792	-5.990	-0.583	0.948	-5.580
115	-0.106	0.792	-5.990	0.010	0.944	-5.580
116	-1.270	0.792	-5.990	-1.250	0.850	-5.610
117	-0.975	0.290	-5.990	-1.150	0.318	-5.600
118	-0.396	0.290	-5.990	-0.439	0.428	-5.600
119	2.060	0.792	-5.990	1.890	0.682	-5.480
120	2.640	0.792	-5.990	2.280	0.806	-5.440
121	1.480	0.792	-5.990	1.600	0.939	-5.530
122	1.770	0.290	-5.990	1.550	0.187	-5.500
123	2.350	0.290	-5.990	1.980	0.322	-5.400
124	-3.430	-0.792	-4.340	-2.850	-0.744	-4.940
125	-2.850	-0.792	-4.340	-2.440	-0.719	-5.340
126	-3.140	-0.290	-4.340	-2.740	-0.062	-5.240
127	-3.720	-0.290	-4.340	-2.930	-0.301	-4.920
128	-4.010	-0.792	-4.340	-3.070	-0.775	-4.770
129	-0.686	-0.792	-4.340	-0.986	-0.927	-5.600
130	-0.106	-0.792	-4.340	0.287	-0.886	-5.570

Table B-1. 6-inch Standard container pipe centroid coordinates

	t = 0 yr			*t* = 1000 yr		
Pipe #	*X* (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
131	-0.396	-0.290	-4.340	-0.281	-0.324	-5.600
132	-0.975	-0.290	-4.340	-1.150	-0.376	-5.530
133	-1.270	-0.792	-4.340	-1.610	-0.753	-5.410
134	2.060	-0.792	-4.340	2.200	-0.862	-5.390
135	2.640	-0.792	-4.340	2.440	-0.648	-5.260
136	2.350	-0.290	-4.340	2.300	-0.432	-5.380
137	1.770	-0.290	-4.340	1.960	-0.534	-5.480
138	1.480	-0.792	-4.340	1.780	-0.890	-5.540
139	-3.430	0.792	-4.340	-2.810	0.637	-4.970
140	-2.850	0.792	-4.340	-2.390	0.664	-5.260
141	-4.010	0.792	-4.340	-3.060	0.888	-4.750
142	-3.720	0.290	-4.340	-2.940	0.190	-4.930
143	-3.140	0.290	-4.340	-2.680	0.283	-5.120
144	-0.686	0.792	-4.340	-0.844	0.706	-5.570
145	-0.106	0.792	-4.340	0.326	0.700	-5.600
146	-1.270	0.792	-4.340	-1.590	0.593	-5.500
147	-0.975	0.290	-4.340	-1.080	0.251	-5.550
148	-0.396	0.290	-4.340	-0.378	0.214	-5.630
149	2.060	0.792	-4.340	2.370	0.683	-5.360
150	2.640	0.792	-4.340	2.560	0.711	-5.240
151	1.480	0.792	-4.340	1.780	0.866	-5.500
152	1.770	0.290	-4.340	2.000	0.041	-5.360
153	2.350	0.290	-4.340	2.420	0.231	-5.310

Table B-2. 12-inch Standard container pipe centroid coordinates

	t = 0 yr			*t* = 1000 yr		
Pipe #	*X* (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
1	-2.057	-0.000	-5.160	-1.807	0.158	-5.323
2	-1.477	-0.000	-5.160	-1.574	0.245	-5.465
3	-1.767	0.502	-5.160	-1.747	0.864	-5.301
4	-2.346	0.502	-5.160	-1.982	0.789	-5.268
5	-2.636	0.000	-5.160	-2.306	-0.132	-5.434
6	-2.346	-0.502	-5.160	-2.067	-0.341	-5.385
7	-1.767	-0.502	-5.160	-1.778	-0.242	-5.373
8	0.686	-0.000	-5.160	0.999	-0.044	-5.539
9	1.265	-0.000	-5.160	1.460	-0.293	-5.348
10	0.975	0.502	-5.160	1.275	0.294	-5.408
11	0.396	0.502	-5.160	0.858	0.825	-5.496
12	0.106	0.000	-5.160	0.304	-0.247	-5.515
13	0.396	-0.502	-5.160	0.874	-0.920	-5.472
14	0.975	-0.502	-5.160	1.264	-0.651	-5.278

Table B-2. 12-inch Standard container pipe centroid coordinates

	t = 0 yr			*t* = 1000 yr		
Pipe #	*X* (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
15	3.428	-0.000	-5.160	2.506	-0.034	-5.228
16	4.007	-0.000	-5.160	2.811	-0.050	-5.137
17	3.718	0.502	-5.160	2.795	0.883	-5.166
18	3.138	0.502	-5.160	2.489	0.511	-5.151
19	2.848	0.000	-5.160	2.178	0.041	-5.263
20	3.138	-0.502	-5.160	2.595	-0.788	-5.330
21	3.718	-0.502	-5.160	2.850	-0.870	-5.197
22	-2.057	-0.000	-5.986	-1.844	0.075	-5.633
23	-1.477	-0.000	-5.986	-1.375	-0.042	-5.597
24	-1.767	0.502	-5.986	-1.669	0.599	-5.627
25	-2.346	0.502	-5.986	-2.042	0.469	-5.586
26	-2.636	0.000	-5.986	-2.154	0.178	-5.604
27	-2.346	-0.502	-5.986	-1.936	-0.642	-5.575
28	-1.767	-0.502	-5.986	-1.488	-0.575	-5.601
29	0.686	-0.000	-5.986	0.465	0.172	-5.519
30	1.265	-0.000	-5.986	1.285	-0.009	-5.604
31	0.975	0.502	-5.986	1.085	0.553	-5.586
32	0.396	0.502	-5.986	0.480	0.669	-5.528
33	0.106	0.000	-5.986	0.088	0.096	-5.655
34	0.396	-0.502	-5.986	0.493	-0.729	-5.602
35	0.975	-0.502	-5.986	0.955	-0.554	-5.510
36	3.428	-0.000	-5.986	2.685	0.027	-5.542
37	4.007	-0.000	-5.986	2.978	0.019	-5.535
38	3.718	0.502	-5.986	2.906	0.527	-5.472
39	3.138	0.502	-5.986	2.598	0.392	-5.469
40	2.848	0.000	-5.986	2.320	0.048	-5.593
41	3.138	-0.502	-5.986	2.525	-0.348	-5.461
42	3.718	-0.502	-5.986	2.879	-0.445	-5.496
43	-2.057	-0.000	-4.334	-2.113	0.016	-5.146
44	-1.477	-0.000	-4.334	-1.436	-0.183	-5.286
45	-1.767	0.502	-4.334	-1.800	0.488	-5.235
46	-2.346	0.502	-4.334	-2.267	0.404	-5.112
47	-2.636	0.000	-4.334	-2.457	0.039	-5.033
48	-2.346	-0.502	-4.334	-2.233	-0.528	-5.079
49	-1.767	-0.502	-4.334	-1.757	-0.614	-5.292
50	0.686	-0.000	-4.334	0.702	0.047	-5.454
51	1.265	-0.000	-4.334	1.234	-0.003	-5.273
52	0.975	0.502	-4.334	0.923	0.431	-5.350
53	0.396	0.502	-4.334	0.294	0.496	-5.469
54	0.106	0.000	-4.334	-0.008	0.128	-5.446
55	0.396	-0.502	-4.334	0.427	-0.455	-5.485
56	0.975	-0.502	-4.334	0.766	-0.365	-5.472
57	3.428	-0.000	-4.334	2.755	0.104	-4.858
58	4.007	-0.000	-4.334	3.044	0.055	-4.698
59	3.718	0.502	-4.334	2.990	0.602	-4.745
60	3.138	0.502	-4.334	2.756	0.487	-4.920
61	2.848	0.000	-4.334	2.482	0.028	-4.934
62	3.138	-0.502	-4.334	2.638	-0.434	-4.965

Table B-2. 12-inch Standard container pipe centroid coordinates

Pipe #	*t* = 0 yr			*t* = 1000 yr		
	X (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
63	3.718	-0.502	-4.334	2.957	-0.491	-4.806
64	-3.428	-0.792	-5.160	-2.517	-0.877	-5.240
65	-2.848	-0.792	-5.160	-2.072	-0.900	-5.179
66	-3.138	-0.290	-5.160	-2.447	-0.438	-5.281
67	-3.718	-0.290	-5.160	-2.756	-0.306	-5.212
68	-4.007	-0.792	-5.160	-2.851	-0.891	-5.122
69	-0.686	-0.792	-5.160	-0.964	-0.951	-5.404
70	-0.106	-0.792	-5.160	-0.027	-0.502	-5.523
71	-0.396	-0.290	-5.160	-0.750	0.038	-5.466
72	-0.975	-0.290	-5.160	-1.145	-0.022	-5.504
73	-1.265	-0.792	-5.160	-1.533	-0.950	-5.401
74	2.057	-0.792	-5.160	2.143	-0.876	-5.507
75	2.636	-0.792	-5.160	2.439	-0.961	-5.433
76	2.346	-0.290	-5.160	2.074	-0.181	-5.348
77	1.767	-0.290	-5.160	1.850	-0.513	-5.399
78	1.477	-0.792	-5.160	1.662	-0.916	-5.303
79	-3.428	0.792	-5.160	-2.560	0.919	-5.251
80	-2.848	0.792	-5.160	-2.216	0.920	-5.223
81	-4.007	0.792	-5.160	-2.862	0.787	-5.124
82	-3.718	0.290	-5.160	-2.897	0.328	-5.159
83	-3.138	0.290	-5.160	-2.516	0.143	-5.348
84	-0.686	0.792	-5.160	-0.959	0.946	-5.427
85	-0.106	0.792	-5.160	-0.018	0.517	-5.512
86	-1.265	0.792	-5.160	-1.433	0.977	-5.419
87	-0.975	0.290	-5.160	-1.341	0.373	-5.521
88	-0.396	0.290	-5.160	-0.512	0.542	-5.463
89	2.057	0.792	-5.160	2.162	0.896	-5.530
90	2.636	0.792	-5.160	2.605	0.940	-5.379
91	1.477	0.792	-5.160	1.683	0.911	-5.357
92	1.767	0.290	-5.160	1.777	0.558	-5.279
93	2.346	0.290	-5.160	2.140	0.447	-5.493
94	-3.428	-0.792	-5.986	-2.669	-0.749	-5.559
95	-2.848	-0.792	-5.986	-2.220	-0.857	-5.509
96	-3.138	-0.290	-5.986	-2.579	-0.282	-5.597
97	-3.718	-0.290	-5.986	-2.954	-0.241	-5.521
98	-4.007	-0.792	-5.986	-2.998	-0.740	-5.554
99	-0.686	-0.792	-5.986	-0.502	-0.805	-5.550
100	-0.106	-0.792	-5.986	0.157	-0.839	-5.512
101	-0.396	-0.290	-5.986	-0.353	-0.183	-5.558
102	-0.975	-0.290	-5.986	-0.776	-0.293	-5.585
103	-1.265	-0.792	-5.986	-1.162	-0.717	-5.536
104	2.057	-0.792	-5.986	1.746	-0.864	-5.605
105	2.636	-0.792	-5.986	2.354	-0.557	-5.574
106	2.346	-0.290	-5.986	1.973	-0.424	-5.623
107	1.767	-0.290	-5.986	1.500	-0.358	-5.585
108	1.477	-0.792	-5.986	1.302	-0.810	-5.529
109	-3.428	0.792	-5.986	-2.697	0.775	-5.563
110	-2.848	0.792	-5.986	-2.240	0.764	-5.588

Table B-2. 12-inch Standard container pipe centroid coordinates

Pipe #	*t* = 0 yr			*t* = 1000 yr		
	X (m)	*Y* (m)	*Z* (m)	*X* (m)	*Y* (m)	*Z* (m)
111	-4.007	0.792	-5.986	-2.997	0.763	-5.555
112	-3.718	0.290	-5.986	-2.842	0.147	-5.578
113	-3.138	0.290	-5.986	-2.493	0.447	-5.505
114	-0.686	0.792	-5.986	-0.498	0.850	-5.561
115	-0.106	0.792	-5.986	0.114	0.853	-5.566
116	-1.265	0.792	-5.986	-1.210	0.715	-5.616
117	-0.975	0.290	-5.986	-0.828	0.464	-5.634
118	-0.396	0.290	-5.986	-0.311	0.224	-5.649
119	2.057	0.792	-5.986	1.744	0.825	-5.615
120	2.636	0.792	-5.986	2.373	0.704	-5.593
121	1.477	0.792	-5.986	1.278	0.790	-5.573
122	1.767	0.290	-5.986	1.528	0.220	-5.497
123	2.346	0.290	-5.986	1.767	0.306	-5.545
124	-3.428	-0.792	-4.334	-2.786	-0.706	-4.844
125	-2.848	-0.792	-4.334	-2.464	-0.793	-4.981
126	-3.138	-0.290	-4.334	-2.711	-0.239	-4.942
127	-3.718	-0.290	-4.334	-3.024	-0.224	-4.741
128	-4.007	-0.792	-4.334	-3.080	-0.792	-4.663
129	-0.686	-0.792	-4.334	-0.833	-0.629	-5.431
130	-0.106	-0.792	-4.334	-0.214	-0.912	-5.471
131	-0.396	-0.290	-4.334	-0.487	-0.431	-5.407
132	-0.975	-0.290	-4.334	-1.115	-0.350	-5.419
133	-1.265	-0.792	-4.334	-1.376	-0.585	-5.306
134	2.057	-0.792	-4.334	2.024	-0.784	-5.133
135	2.636	-0.792	-4.334	2.458	-0.788	-5.028
136	2.346	-0.290	-4.334	2.269	-0.458	-5.070
137	1.767	-0.290	-4.334	1.778	-0.206	-5.213
138	1.477	-0.792	-4.334	1.564	-0.607	-5.173
139	-3.428	0.792	-4.334	-2.757	0.787	-4.870
140	-2.848	0.792	-4.334	-2.493	0.709	-4.987
141	-4.007	0.792	-4.334	-3.082	0.790	-4.668
142	-3.718	0.290	-4.334	-3.052	0.286	-4.678
143	-3.138	0.290	-4.334	-2.701	0.304	-4.935
144	-0.686	0.792	-4.334	-0.853	0.671	-5.357
145	-0.106	0.792	-4.334	-0.223	0.913	-5.436
146	-1.265	0.792	-4.334	-1.318	0.597	-5.254
147	-0.975	0.290	-4.334	-1.001	0.310	-5.382
148	-0.396	0.290	-4.334	-0.376	0.189	-5.376
149	2.057	0.792	-4.334	2.019	0.823	-5.180
150	2.636	0.792	-4.334	2.408	0.780	-5.031
151	1.477	0.792	-4.334	1.323	0.785	-5.252
152	1.767	0.290	-4.334	1.809	0.139	-5.179
153	2.346	0.290	-4.334	2.199	0.355	-5.075

DISTRIBUTION

Email—External (encrypt for OUO)

Name	Company Email Address	Company Name
Brett Brickner	bricknerbd@ornl.gov	Oak Ridge National Laboratory
Ellen Saylor	saylorem@ornl.gov	Oak Ridge National Laboratory
John Scaglione	scaglionejm@ornl.gov	Oak Ridge National Laboratory

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
T. Shelton	1542	trshelt@sandia.gov
S. Klenke	1550	seklenk@sandia.gov
B. Owens	1554	bcowens@sandia.gov
C. Vignes	1554	cvignes@sandia.gov
J. Koester	1555	jkoeste@sandia.gov
K. Mish	1555	kdmish@sandia.gov
J. Rath	1555	jsrath@sandia.gov
J. Pott	1557	jpott@sandia.gov
E. Fang	1558	hefang@sandia.gov
R. C. Camphouse	8842	rccamph@sandia.gov
E. Matteo	8842	enmatte@sandia.gov
R. Rechard	8842	rprecha@sandia.gov
E. Hardin	8844	ehardin@sandia.gov
K. Kuhlman	8844	klkuhlm@sandia.gov
M. Mills	8844	mmmills@sandia.gov
E. Stein	8844	ergiamb@sandia.gov
B. Park	8862	bypark@sandia.gov
S. Sobolik	8862	srsobol@sandia.gov
T. Zeitler	8862	tzeitle@sandia.gov
S. Broome	8864	stbroom@sandia.gov
C. Herrick	8864	cgherri@sandia.gov
R. Jensen	8864	rpjense@sandia.gov
S. Bauer	8866	sjbauer@sandia.gov
P. Shoemaker	8880	peshoem@sandia.gov
B. Day	8881	baday@sandia.gov
D. Kicker	8881	dckicke@sandia.gov
R. Kirkes	8883	grkirke@sandia.gov
G. Duran	8883	gasosa@sandia.gov
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories