This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2023-10477

SANDIA REPORT

SAND2023-10477
(Supersedes SAND2019-3106 O)
Printed January, 2023

Sandia National Laboratories

Simulations of Criticality Control Overpack Container Compaction at the Waste Isolation Pilot Plant

Benjamin Reedlunn, James Bean, John Wilkes, and John Bignell

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

Sandia National Laboratories
U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods



ABSTRACT

Criticality Control Overpack (CCO) containers are being considered for the disposal of defense-related nuclear waste at the Waste Isolation Pilot Plant (WIPP). At WIPP, these containers would be placed in underground disposal rooms, which will naturally close and compact the containers closer to one another over several centuries. This report details simulations to predict the final container configuration as an input to nuclear criticality assessments. Each container was discretely modeled, including the plywood and stainless steel pipe inside the 55-gallon drum, in order to capture its complex mechanical behavior. Although these high-fidelity simulations were computationally intensive, several different material models were considered in an attempt to reasonably bound the horizontal and vertical compaction percentages. When exceptionally strong materials were used for the containers, the horizontal and vertical closure respectively stabilized at 43.9 % and 93.7 %. At the other extreme, when the containers completely degraded and the clay seams between the salt layers were glued, the horizontal and vertical closure reached respective final values of 48.6 % and 100 %.

ACKNOWLEDGMENT

The authors would like to acknowledge Ernest Hardin and Brad Day for their helpful suggestions and for reviewing a draft of this report. We are also grateful to Timothy Shelton and Matthew Mosby for their troubleshooting advice, to Bradley Salzbrenner for sharing his tensile data on carbon steel from a 55-gallon drum, and to Paul Shoemaker for being patient as we developed a method to run these simulations.

PREFACE

This report supersedes Reedlunn et al. (2019) (SAND2019-3106 O) with the following four small changes:

1. A citation of Saylor and Scaglione (2018) was added to Chapter 1 to provide further motivation to the work contained herein.
2. The pipe diameter-to-length and diameter-to-thickness ratios quoted in Chapter 1 were updated to use more accurate pipe dimensions.
3. The presentation of the clay seam model in Section 2.1.4.1 was revised to note that the Clay F and G friction coefficients were inadvertently set to 0.5, instead of 0.2 or 0.0, in some simulations.
4. The discussion of Eq. (A.1) was modified to indicate that Sierra/SM actually normalized the residual by the norm of the reaction forces, not the norm of the externally applied forces.

These changes do not impact the original results or conclusions in Reedlunn et al. (2019), so those topics remain unchanged in this report.

After Reedlunn et al. (2019) was released, several follow-on container compaction studies were released (Reedlunn and Bean 2019; Reedlunn and Bean 2020a; Reedlunn and Bean 2021) to support nuclear criticality assessments. Note that these studies strove to construct representative container compaction scenarios rather than worst case compaction scenarios because, as discussed in Rechard et al. (2022, Section VIII.A), screening out nuclear criticality from assessments of WIPP's performance is based on analyses of representative conditions. These compaction studies considered 6-inch Pipe Overpack (POP) containers and 12-inch POP containers, in addition to Criticality Control Overpack (CCO) containers. The studies compared how the different containers compacted, looked at container compaction due to two different roof fall scenarios, investigated the sensitivity to clay seam behavior, compared container compaction in lower vs. upper horizon disposal rooms, and explored the sensitivity to container emplacement configuration. Furthermore, the studies post-processed simulation results in more sophisticated ways: the room closure results were converted into the compaction of an idealized uniform array, the centerline of each pipe was tracked and visualized, and the simulations were compared using a physically-based pipe concentration metric. These comparisons showed the CCO and 6-inch POP containers reached similar pipe concentration distributions, while the 12-inch POP container cases, which have roughly double the stainless steel, exhibited 10 % smaller average pipe concentrations. The other variations to the simulation inputs, however, had relatively minor effects on the final compacted configuration, which suggests substantial changes in simulation inputs would be needed to induce significant changes in the average pipe concentrations.

CONTENTS

1. Introduction **13**

2. Model Setup **18**
2.1. Geomechanical Model 18
2.1.1. History of Model Development 18
2.1.2. Geometry and Boundary Conditions 18
2.1.3. Material Models 21
2.1.4. Interface Models 23
2.2. Container Model 25
2.2.1. Geometry and Boundary Conditions 25
2.2.2. Material Models 26
2.2.3. Interface Models 36
2.3. Numerical Approach 36
2.3.1. Potential Approaches 36
2.3.2. Viscoplastic Strain Rate Scaling 37

3. Results **40**
3.1. Sensitivity to Container Behavior 40
3.1.1. Elastic 40
3.1.2. Elastic-Plastic 42
3.1.3. Elastic-Plastic-Fracture 42
3.1.4. Strong-Elastic-Plastic-Fracture 44
3.1.5. Quantitative Comparisons 44
3.2. Sensitivity to Clay Seam Behavior 48

4. Summary and Potential Future Work **50**

References **53**

Bibliography **53**

Appendices **57**

A. Numerical Settings **57**
A.1. Implicit Quasi-statics 57
A.2. Explicit Dynamics 59
A.2.1. Viscoplastic Rate Scaling 60
A.2.2. Number of Contact Iterations 62
A.2.3. Element Size 63

A.2.4. Density Scaling 64
A.3. Finite Element Meshes 66

B. Preliminary Validations of the Geomechanical Model **68**

C. Container Data **73**
C.1. Technical Drawings 73
C.2. Mechanical Response of the Carbon Steel 77
C.2.1. Experimental Setup 77
C.2.2. Experimental Results 77
C.2.3. Hardening Curve Calibration Procedure 81

D. Simulation Files **82**
D.1. Comma Separated Variable Files 82
D.1.1. Stratigraphy 82
D.2. Cubit Journal Files 82
D.2.1. Disposal Room 82
D.2.2. Container 90
D.2.3. Container Emplacement 95
D.3. Sierra Input Files 101
D.3.1. Empty Disposal Room Simulations 101
D.3.2. Container Compaction Simulations 119

LIST OF FIGURES

Figure 1-1. Standard WIPP container 13
Figure 1-2. Container configuration immediately after disposal and the corresponding homogenized idealization. 14
Figure 1-3. Simulated compaction of homogenized waste. The left side is a mirror boundary condition. (Fig. 22 from Park and Hansen (2005)) 15
Figure 1-4. Criticality Control Overpack container. 16

Figure 2-1. Idealized Stratigraphy 19
Figure 2-2. Geomechanical simulation geometry and boundary conditions in the X-Z plane. 21
Figure 2-3. Idealized container geometry. (Quarter wedge removed for visualization purposes.) 25
Figure 2-4. (a) Attempted container configuration that exhibited penetration issues, and (b) adapted container configuration that was utilized in the compaction simulations. 26
Figure 2-5. True stress vs. true strain curves under uniaxial tension for container materials . 28
Figure 2-6. Carbon steel hardening curves 31
Figure 2-7. Stainless steel hardening curve 33

Figure 3-1. Predictions of elastic and elastic-plastic container deformation at selected instances in time. (Geology and 55-gallon drums removed for visualization purposes.) 41
Figure 3-2. Von Mises stress distribution in stainless steel pipes for elastic containers at $t = 500$ yr. 42
Figure 3-3. Predictions of elastic-plastic-fracture and strong-elastic-plastic-fracture container deformation at selected instances in time. (Geology and 55-gallon drums removed for visualization purposes.) 43
Figure 3-4. Room closure with five different container behaviors 45
Figure 3-5. Empty disposal room shapes at various times for the three clay seam behaviors studied. 47
Figure 3-6. Empty room closure with three different clay seam behaviors. 48

Figure A-1. Justification for neglecting steady state mechanism 3 in further implicit quasi-static simulations. 58
Figure A-2. Maximum relative residual norm $\hat{R}_{\max}$ convergence study. 59
Figure A-3. Viscoplastic scaling curves. 60
Figure A-4. Room closure and kinetic energy over internal energy histories for different viscoplastic scalings, compared against an implicit quasi-static simulation. (Note that E_k/E_i curve for $s_1 = 2\times10^{11}$ s^{-1} was likely under-sampled during $0 \le t \le 2$ yr.) 61

Figure A-5. Room closures histories for different numbers of explicit dynamic momentum balance iterations. 63
Figure A-6. Coarsest and finest finite element meshes utilized in geomechanics mesh convergence study. 64
Figure A-7. Convergence of horizontal and vertical closure histories as the element size is decreased. 65
Figure A-8. Finite element meshes utilized in container compaction simulations. 67

Figure B-1. Locations of Rooms D, G, and Q at the WIPP 68
Figure B-2. Idealized plane strain geometry of Rooms D, G, and Q. 69
Figure B-3. Room D finite element mesh. 70
Figure B-4. Preliminary validations of the geomechanical model 72

Figure C-1. Criticality Control Overpack Drawing, Sheet 1 (McGonagill 2015) 74
Figure C-2. Criticality Control Overpack Drawing, Sheet 2 (McGonagill 2015) 75
Figure C-3. 55-Gallon Drum Detail Drawing (Skolnik Industries 2018) 76
Figure C-4. Carbon Steel Specimen Locations and Dimensions (Salzbrenner and Smith 2018) 78
Figure C-5. Carbon steel mechanical response curves and engineering strain histories 79
Figure C-6. Photos of drum lid specimen fractures. Left and right images respectively show the thickness and width reductions at the fracture site. (Salzbrenner and Smith 2018) 80

LIST OF TABLES

Table 2-1. Anhydrite and Polyhalite Calibrations. 23
Table 2-2. Intended Clay Seam Parameter Sets 24
Table 2-3. Actual Clay Seam Parameter Sets Used in Disposal Room Simulations 24
Table 2-4. Carbon steel hardening curve data points 30
Table 2-6. Carbon Steel Calibrations 32
Table 2-7. Stainless steel hardening curve data points 32
Table 2-9. Stainless Steel Calibrations 33
Table 2-10. Typical stiffness and strength measurements on pine plywood 34
Table 2-11. Plywood Calibrations 35

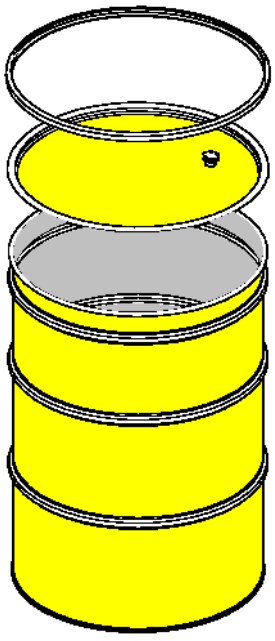
Table 3-1. Final closure predictions 46

Table B-1. Rooms D, G, and Q dimensions (in meters). 71

Table C-1. Carbon Steel Specimen Dimensions 77

1. INTRODUCTION

The Waste Isolation Pilot Plant (WIPP) is an operating geologic repository for defense-related nuclear waste. The waste arrives at WIPP in various types of containers and a new container, called a Criticality Control Overpack (CCO) container, is under consideration. A CCO container can contain up to 380 fissile gram equivalent (FGE) of plutonium per container, which is 180 FGE above the current limit of 200 FGE per container, so implications for nuclear criticality are being carefully examined. Once waste containers are placed in an underground disposal room, the natural process of room closure will encapsulate and compact the containers over decades to centuries. A previous CCO criticality analysis (Saylor and Scaglione 2018) made very conservative assumptions about CCO compaction in the absence of a citable reference on how CCOs might realistically be compacted by room closure. This report fills that void by presenting several high-fidelity geomechanical simulations of CCO compaction within a disposal room that will hopefully be useful in future criticality analyses.



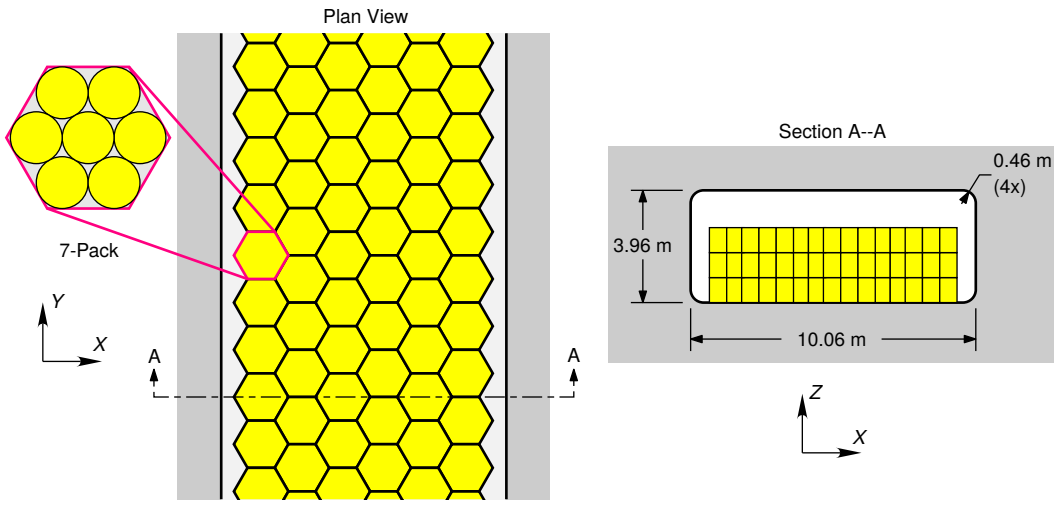
(a) Schematic (Park and Hansen 2005)



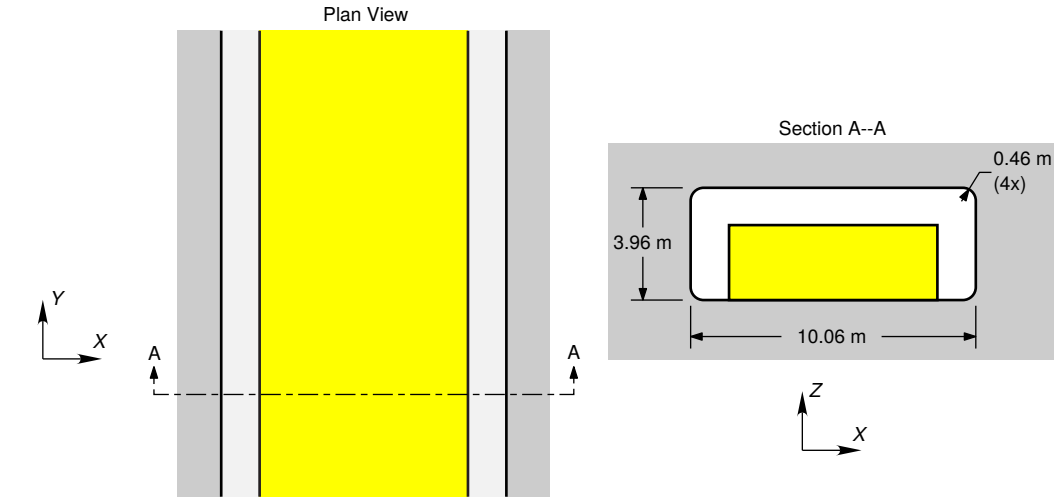
(b) Photo (with Surrogate Waste) (Broome et al. 2016)

Figure 1-1. Standard WIPP container

Past simulations of container compaction at the WIPP have homogenized the containers into a solid, spatially uniform, material. As shown in Fig. 1-1, standard waste containers, the most



(a) Containers upon emplacement



(b) Homogenized idealization of Fig. 1-2a

Figure 1-2. Container configuration immediately after disposal and the corresponding homogenized idealization.

common type of container at the WIPP, are 55-gallon drums filled with all types of contaminated debris: cellulosics, metals, sorbants, and polymers*. These containers are typically bundled together in hexagon shaped 7-packs: six containers surround one container in the center. Upon emplacement in a WIPP disposal room, the 7-packs are stacked up to three-high and arranged in a hexagonal configuration (see Fig. 1-2a). For geomechanical modeling, however, the complexity in Fig. 1-2a is typically replaced with the idealized geometry in Fig. 1-2b. Instead of modeling every individual container, all the containers are represented by a single, isotropic, material (see, for example, Stone (1997a), Park and Hansen (2005), and Park and Holland (2007)). Fig. 1-3

*See Broome et al. (2014) and Broome et al. (2016) for hydrostatic, uniaxial, and triaxial compression tests on these standard containers filled with surrogate non-degraded and degraded waste.

depicts six deformed mesh images from one such simulation. 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 at 25 years and slowly compresses it over the next 10,000 years. As expected, the waste deformation is relatively uniform. Park and Hansen (2005) also considered the impact of mixing standard containers with other container types, such as pipe overpack containers, but each container type was still modeled as a homogeneous, spatially uniform, material.

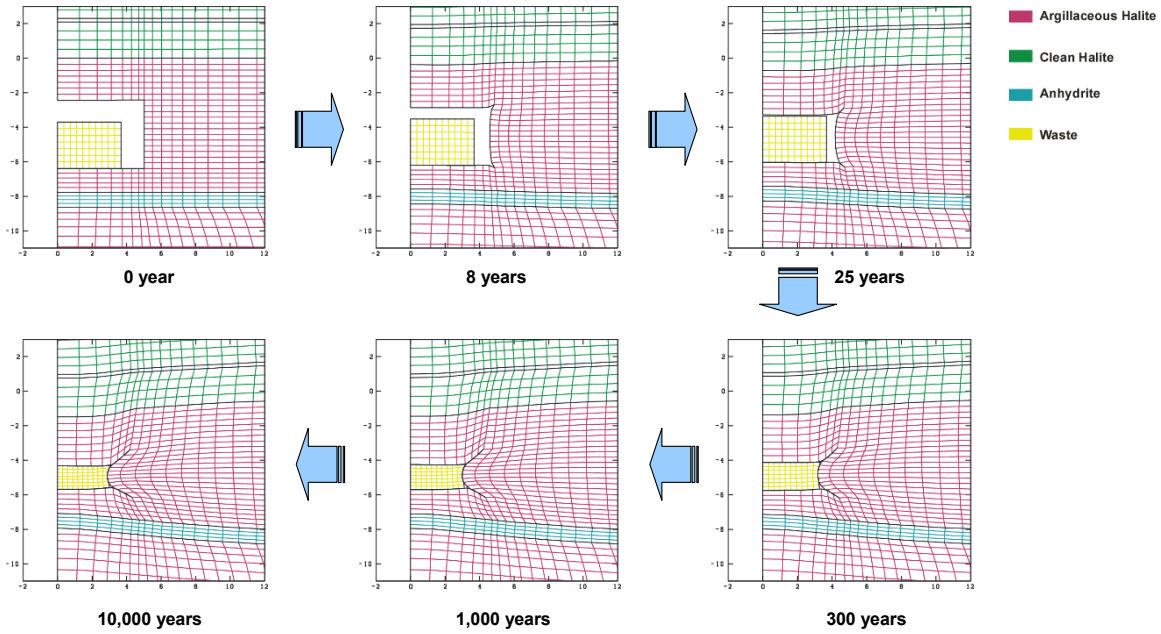
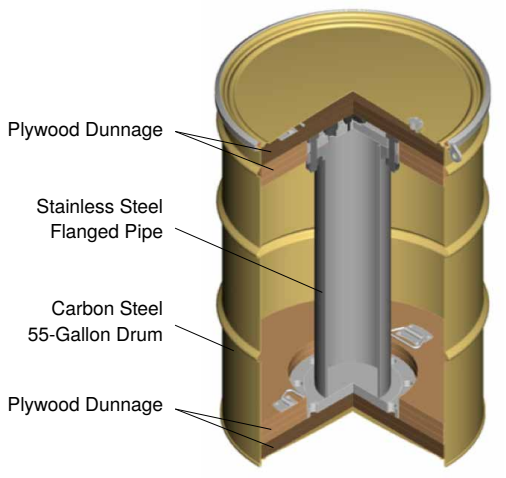


Figure 1-3. Simulated compaction of homogenized waste. The left side is a mirror boundary condition. (Fig. 22 from Park and Hansen (2005))

The new criticality control overpack (CCO) containers differ from previously used waste disposal containers. A schematic and a photo of a CCO container are shown in Fig. 1-4 and detailed drawings can be found in Figs. C-1 and C-2. A CCO consists of a stainless steel flanged pipe sandwiched between plywood dunnage layers inside a carbon steel 55-gallon drum. The waste will be placed inside the stainless steel pipe, which is also called a criticality control container (CCC). Note that the volume between the 55-gallon drum and the pipe is filled only with air (see Fig. 1-4b), making the container far more stiff in the axial direction than the radial direction. Furthermore, the pipe has a nominal diameter D_p to thickness B_p ratio of $D_p/B_p = 23.7$ and a length L_p to diameter ratio of $L_p/D_p = 4.0$. These dimensions make the pipe thin and slender enough to be susceptible to shell and/or column buckling.

The pipe overpack container (POC) is the most similar preexisting container design to a CCO container. A POC also consists of a stainless steel flanged pipe sandwiched between plywood discs inside a carbon steel 55-gallon drum. Instead of air between the pipe and the 55-gallon drum, a POC has Celotex© fiberboard. This fiberboard is meant to protect against high-speed impacts during handling and transportation, but it probably provides little structural resistance at the slow strain rates applied during room closure. Several different POC design variations exist,



(a) Solid model (quarter wedge removed) (modified from McGonagill (2015))



(b) Photo (lid removed) (Incorporated 2011)

Figure 1-4. Criticality Control Overpack container.

yet all have one of two different pipes. The 12-inch diameter pipe design has non-dimensional ratios of $D_p/B_p = 51.0$ and $L_p/D_p = 2.0$, while the 6-inch pipe design has ratios of $D_p/B_p = 23.7$ and $L_p/D_p = 3.9$. The 6-inch pipe design is similar to the stainless steel pipe inside a CCO container.

Park and Hansen (2004) and Park and Hansen (2005) elected to homogenize the standard 6 and 12-inch POCs in order to perform container compaction simulations. In the absence of experimental measurements, Park and Hansen (2004) first modeled the drum, pipe, plywood, and Celotex© using assumed material properties from Ludwigsen et al. (1998). They then applied hydrostatic, triaxial, and uniaxial compression to an axisymmetric model of a single container to generate three different response curves. These response curves were used to fit an isotropic volumetric plasticity model. Finally, Park and Hansen (2005) assumed the homogenized behavior of one container was the same as the homogenized behavior of the hundreds of containers represented by the yellow waste rectangle shown in Fig. 1-2b.

The Park and Hansen (2004) and Park and Hansen (2005) homogenization approach was reasonable given the computational resources at the time, but it likely produced an overly stiff constitutive model for the waste. As containers are actually compacted by the closing room, one can imagine that two or more containers could buckle in unison, interacting as they deform. Actual containers may also slide past one another such that a collection of containers has very little resistance to shear along certain directions. By modeling a single axisymmetric container, Park and Hansen (2004) and Park and Hansen (2005) did not permit buckling wavelengths that span multiple containers and non-axisymmetric buckling modes, such as column buckling. Furthermore, their model essentially glued the 55-gallon drums to each other and did not capture the container's structural anisotropy.

A different homogenization scheme was considered here, but discarded in favor of discretely modeling each CCO container. The homogenization scheme considered was to model a

representative volume of multiple containers and continue increasing the volume (adding containers) until the mechanical response converged for several characteristic deformation modes. One would need to chose those deformation modes, select an appropriate homogenized model, and then fit it to the mechanical responses. This homogenization procedure would have to be repeated any time one wished to change the material properties of the container materials. In addition, it is possible that the converged representative volume may be nearly the height or width of the disposal room, meaning that scale separation between the room deformation and the container deformation would not exist. Discretely modeling each container avoids homogenization error associated with replacing a high-fidelity model with a reduced order model, allows one to avoid the labor intensive homogenization procedure, allows one to change container material models at will, and captures the interactions between the room deformed shape and the container deformation. Discrete container modeling obviously comes with a significant increase in computational cost, but processing power has also advanced significantly since Park and Hansen (2004) performed their studies. In addition, a new numerical technique was developed herein to utilize Sierra/Solid Mechanics's explicit dynamics capability instead of implicit quasi-statics. (Explicit dynamics is far more robust than implicit quasi-statics when simulating contact, severe deformations, and failing materials.) It may be worth creating a homogenized model for collections of CCOs in the future, but a higher fidelity model will still be useful as a reference for the homogenization error.

Although the simulations presented herein are certainly more involved than those in the past, they are still missing a significant feature: they do not include fracture of the rock salt or anhydrite. If left unrestrained, fractures above rooms can cause large slabs of rock to fall into the room and fractures beneath the floor can cause substantial floor heave. Examples with photos of actual events at the WIPP are discussed in Section 2 of Kicker et al. (2017). Rock falls might crush some containers and scatter other containers around the room. These events could potentially lead to clusters of containers (high plutonium density) in some regions and large regions of salt between containers (low plutonium density) in other regions. Unfortunately, fracturing is a difficult processes to simulate in the finite element method. It causes pathological mesh dependence, where simulation results do not converge with mesh refinement. A host of numerical techniques have been developed to alleviate this issue over the last several decades, but these techniques are still topics of research. Constitutive models for rock salt fracturing (and healing) are also topics of research, and the existing models in the literature have not been implemented into Sierra/SolidMechanics (2018). For these reasons, the impact of fracturing is left for future work. Even without rock falls, the predictions herein are still useful estimates of the container compaction averaged over the volume of the room.

The remainder of this report is organized as follows. Chapter 2 discusses the geomechanical model, container model, and the numerical approach that enabled explicit dynamic simulations of long duration events. Chapter 3 presents the container compaction simulation results for a few different container material model behaviors and clay seam behaviors. Chapter 4 completes the report with a summary.

2. MODEL SETUP

This chapter defines the geomechanical model (Section 2.1), container model (Section 2.2), and numerical approach (Section 2.3) utilized in the container compaction simulations. Although geomechanicians often treat compressive stresses and strains as positive, these simulations also include the mechanical behavior of metal components. Consequently, tensile stresses and strains are considered positive herein, according to standard solid mechanics conventions.

2.1. Geomechanical Model

2.1.1. History of Model Development

Sandia's geomechanical model for room closure at the WIPP has evolved considerably over the years. The early efforts to simulate room closure in the mid-1980's under-predicted vertical closure rates at the WIPP by roughly 3.5× (see Fig. 7 in Munson et al. (1987), for example) causing investigators to explore a series of changes to the way room closure was modeled. Munson et al. (1989) were able to match the measured closure of Room D and other in-situ experiments at the WIPP (Munson 1997), but several model parameters were tuned for engineering purposes (Reedlunn 2016). Nevertheless, much of the Munson et al. (1989) simulation setup became the accepted procedure for modeling room closure at the WIPP. More recently, the Munson-Dawson (MD) model for rock salt was re-calibrated against a series of new laboratory tests on rock salt from the WIPP (Reedlunn 2016). This laboratory-based MD model calibration under-predicted the Room D vertical closure rate by 3.5× at 7.8 years (see Cal 1B prediction in Fig. 5a of Reedlunn (2018b)), similar to the mid-1980's predictions. Subsequent improvements to the MD model (Reedlunn 2018a) significantly reduced the Room D vertical closure rate under-prediction to 1.3× at 7.8 years, and produced reasonable predictions of Room G and Room Q's closure rates, as shown in Appendix B. This latest geomechanical model is utilized herein because it has the strongest scientific foundation.

2.1.2. Geometry and Boundary Conditions

Figure 2-1 depicts the stratigraphy used in the simulations. Layers are color coded as to whether they are salt, anhydrite, or polyhalite. Some layers have clay seams between them, marked with brown lines. All elevations are referenced off of clay seam G, which is about 648 m beneath the ground surface.

Figure 2-2 exhibits the two-dimensional simulation domain, boundary conditions, and select dimensions. The disposal room's 91.44 m length in the Y-direction (into the page) was assumed

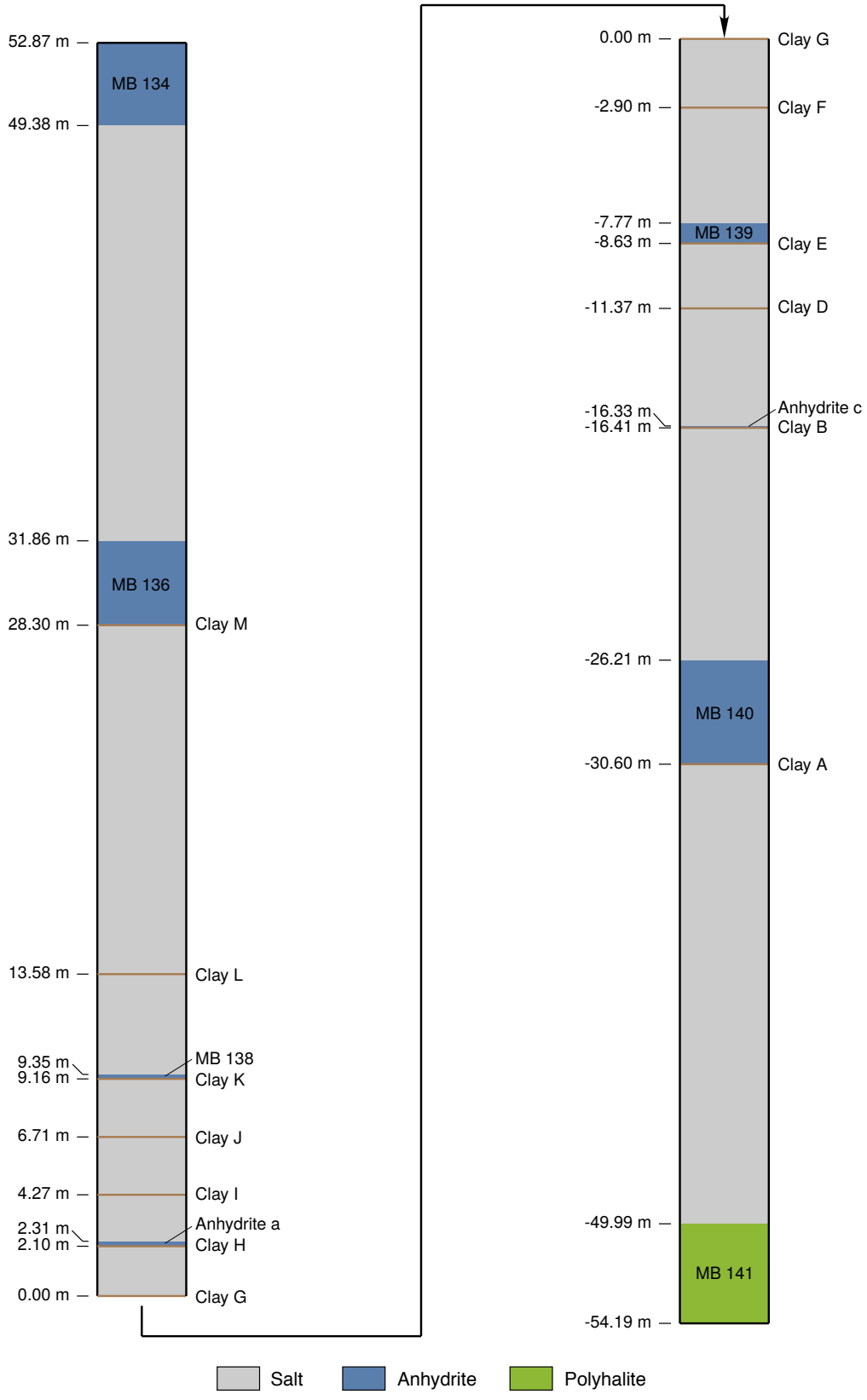


Figure 2-1. Idealized Stratigraphy

to be long enough relative to the height and width of the room to be treated as plane strain. The top of the room coincided with Clay seam G, which is the case in disposal panels 3, 4, 5, and 6 at the WIPP. (Compaction of CCO containers on the repository lower horizon (panels 1, 2, 7, and 8) is left for future work.) The as-excavated room width and height were $L_{\mathrm{h}} = 10.06$ m and $L_{\mathrm{v}} = 3.96$ m, respectively. The room corners were made square, instead of rounded, in order to avoid severe element distortion as the room closed. Each disposal panel at the WIPP consists of seven rooms placed 40.54 m apart, east to west. Here, the seven disposal rooms were treated as if they were an infinite array, such that only one disposal room with mirror boundary conditions $D_{\mathrm{h}} = 40.54$ m apart needed to be modeled. (See the 1991 memorandum of C. M. Stone and J. G. Argüello (pg. A-121 to A-134 of Butcher and Mendenhall (1993)) for a comparison of an infinite array model against a model that explicitly represented all seven rooms in a panel.) Although the geomechanical model also had left-right symmetry, that symmetry was broken once the containers inside the room buckled*. Therefore, container compaction simulations included both the left and right sides of the domain, while the left half of the domain was replaced with a mirror boundary condition for empty room simulations. The top and bottom of the simulation domain were assumed to be far enough away from the room ($D_{\mathrm{v1}} = 52.87$ m and $D_{\mathrm{v2}} = -54.19$ m) to not impact the room closure. 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 $\delta_{\mathrm{v}}^{\mathrm{roof}}$ are shown in Fig. 2-2. These were used to define the horizontal and vertical closure of the room as $\delta_{\mathrm{h}} = \delta_{\mathrm{h}}^{\mathrm{left}} + \delta_{\mathrm{h}}^{\mathrm{right}}$ and $\delta_{\mathrm{v}} = \delta_{\mathrm{v}}^{\mathrm{floor}} + \delta_{\mathrm{v}}^{\mathrm{roof}}$. All material points were initialized with a lithostatic stress state that varied linearly from $p_{\mathrm{top}} = 13.57$ MPa at the domain top to $p_{\mathrm{bot}} = 15.97$ MPa at the domain bottom. This lithostatic pressure gradient is consistent with a uniform rock density of $\rho = 2300$ kg/m^3 and an applied gravity body force of $g = 9.79$ m/s^2. The rock mass temperature was spatially uniform at $T = 300$ K.

The room was introduced into the simulation almost instantaneously as a void at time $t = t_0 = 0$, rather than modeling the excavation process. True instantaneous excavation was not possible because it would have made implicit quasi-static convergence difficult, and would have caused inertial effects in explicit dynamic simulations. Instead, the excavation was spread out over multiple time steps by applying a time-varying fluid pressure to the surfaces of the room. At the very start of the simulation ($t = t_{-1} < t_0$), the fluid pressure varied linearly from the top to the bottom of the room according to the lithostatic pressure that existed prior to room excavation. Then the fluid mass density ρ_{fluid} was reduced from the salt mass density ρ to zero according to

$$\rho_{\mathrm{fluid}} = \frac{\rho}{2}\left[1 - \cos\left(\frac{\pi t}{t_0 - t_{-1}}\right)\right], \tag{2.1}$$

between $t = t_{-1}$ and $t = t_0$. The salt responded virtually elastically to this fluid density reduction because $t_{-1} = -10$ ms and $t_0 = 0$ s. Once $t > t_0$, the fluid density was held to zero.

Past simulations of container compaction, such as those in Stone (1997a), 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 over a wide range to capture the uncertainty in the rate of gas generation and

*It may be worthwhile to enforce left-right symmetry in the future, even though it would constrain container buckling, in order to free up computational resources.

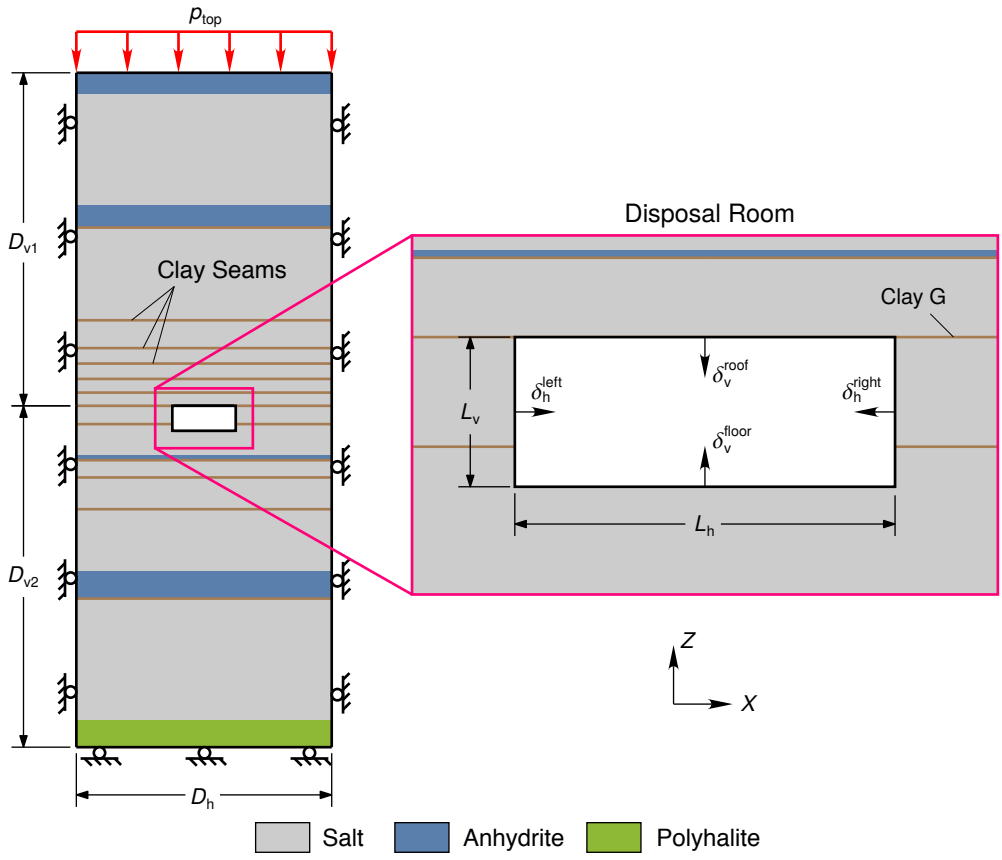


Figure 2-2. Geomechanical simulation geometry and boundary conditions in the X-Z plane.

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 container compaction and neglecting it should not significantly affect the vertical-to-horizontal closure ratio δ_v/δ_h.

2.1.3. Material Models

2.1.3.1. Salt

The salt's mechanical behavior was simulated using the `md_viscoplastic` implementation of the Munson-Dawson (MD) model. The MD model was originally defined in Munson and Dawson (1979), Munson and Dawson (1982), and Munson et al. (1989), but, as mentioned in Section 2.1.1, Reedlunn (2018a) recently added a few new features. MD model Calibration 3B from Table 5.2 in Reedlunn (2018a) was utilized herein.

2.1.3.2. Anhydrite and Polyhalite

The anhydrite and polyhalite were modeled with a hypoelastic, perfectly plastic, Drucker-Prager model. In this model, the strain rate is decomposed as

$$\dot{\boldsymbol{\varepsilon}} = \dot{\boldsymbol{\varepsilon}}^{\mathrm{el}} + \dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}}, \tag{2.2}$$

where $\dot{\boldsymbol{\varepsilon}}^{\mathrm{el}}$ and $\dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}}$ represent the elastic and plastic strain rates, respectively. The elastic portion of the model utilizes the following simple linear relationship between the elastic strain rate $\dot{\boldsymbol{\varepsilon}}^{\mathrm{el}}$ and the stress rate $\dot{\boldsymbol{\sigma}}$,

$$\dot{\boldsymbol{\sigma}} = C : \dot{\boldsymbol{\varepsilon}}^{\mathrm{el}} = C : \left(\dot{\boldsymbol{\varepsilon}} - \dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}}\right) \tag{2.3}$$

$$C = (B - 2/3\,\mu)\,\boldsymbol{I} \otimes \boldsymbol{I} + 2\mu\,\mathcal{I}, \tag{2.4}$$

where C is the fourth-order isotropic elastic stiffness tensor composed of the bulk modulus B, the shear modulus μ, the second-order identity tensor $\boldsymbol{I}$, and the forth-order symmetric identity tensor $\mathcal{I}$. The plastic strain $\boldsymbol{\varepsilon}^{\mathrm{pl}}$ evolves when the Drucker-Prager yield function $g = 0$. Let $I_1 = \mathrm{tr}(\boldsymbol{\sigma})$ be the first invariant of the stress tensor $\boldsymbol{\sigma}$, and let $J_2 = \boldsymbol{\sigma}^{\mathrm{dev}} : \boldsymbol{\sigma}^{\mathrm{dev}}/2$ be the second invariant of the deviatoric stress tensor $\boldsymbol{\sigma}^{\mathrm{dev}} = \boldsymbol{\sigma} - I_1/3\,\boldsymbol{I}$. With these definitions, the Drucker-Prager yield function g can be expressed in terms of the material constants a and C as

$$g = \sqrt{J_2} + a\,I_1 - C, \tag{2.5}$$

keeping in mind that tensile stresses are positive here.

When $g = 0$, the plastic flow potential determines the ratio of the plastic flow components, in other words, the plastic flow direction. Although many publications (see H. S. Morgan and Krieg (1984), Munson and H. Morgan (1986), and Stone (1997a) for example) utilize this Drucker-Prager model for the anhydrite and polyhalite, none appear to document the plastic flow potential. Fortunately, the SANTOS input file in Appendix A of Stone (1997a) shows that the `soils_and_crushable_foams` model was utilized for the anhydrite and polyhalite. According to the SANTOS and LAMÉ manuals (Stone 1997b; Scherzinger and Lester 2018), the `soils_and_crushable_foams` model uses a von Mises flow potential

$$\psi = \sqrt{3\,J_2}, \tag{2.6}$$

and a non-associative flow rule

$$\dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}} = \lambda \frac{\partial \psi}{\partial \boldsymbol{\sigma}}, \tag{2.7}$$

where λ is the consistency parameter.

The `soils_and_crushable_foams` model was used for the polyhalite herein, but not for the anhydrite. As discussed in Section 2.2 of Reedlunn (2016), strain localizations appeared in the anhydrite "a" layer when Room D closure simulations were attempted with the `soils_and_crushable_foams` model for the anhydrite. These strain localizations prevented the implicit solver from converging, so an investigation ensued. The `kayenta` model in Sierra/Solid Mechanics was calibrated to replicate the `soils_and_crushable_foams` model calibration, and

it also exhibited the strain localizations. The `kayenta` model, however, allows one to adjust the Drucker-Prager dilatation angle to transition from a von Mises flow potential to an associative flow potential. Rudnicki and Rice (1975) showed that non-associative flow rules can lead to material instabilities, and the strain localizations in the Room D simulations disappeared when an associative flow rule was used for the anhydrite. Thus, the `kayenta` model with an associative flow rule,

$$\dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}} = \lambda \frac{\partial g}{\partial \boldsymbol{\sigma}}, \tag{2.8}$$

was used in Reedlunn (2016) and this work continues to use the same `kayenta` model calibration for the anhydrite.

The anhydrite and polyhalite calibrations specified in H. S. Morgan and Krieg (1984) are listed in Table 2-1 for reference. The mass density is ρ. The Sierra/Solid Mechanics input syntax can be found in Appendix D.3.

Table 2-1. Anhydrite and Polyhalite Calibrations.

Parameter	Units	Anhydrite	Polyhalite
μ	GPa	27.8	20.3
B	GPa	83.4	65.8
a	–	0.45	0.473
C	MPa	1.35	1.42
ρ	kg/m^3	2300	2300

2.1.4. Interface Models

2.1.4.1. Clay Seams

The clay seams were represented as sliding contact surfaces. The lateral sliding was modeled using Coulomb friction,

$$\boldsymbol{t}_{\mathrm{t}} = \mu_{\mathrm{c}} \, \boldsymbol{t}_{\mathrm{n}}, \tag{2.9}$$

where $\boldsymbol{t}_{\mathrm{t}}$ and $\boldsymbol{t}_{\mathrm{n}}$ respectively are the tangential and normal traction at the interface and μ_{c} is the friction coefficient for the clay seams. Clay seams are assumed to be infinitely stiff normal to the seam plane, but they can separate if the normal stress exceeds the seam's tensile strength σ_{c}.

Although work to characterize clay seam behavior is ongoing (Sobolik 2017), the friction coefficient μ_{c} and tensile strength σ_{c} of WIPP clay seams have never been experimentally measured. In the absence of measurements, Munson et al. (1989) chose nominal values of $\mu_{\mathrm{c}} = 0.2$ and $\sigma_{\mathrm{c}} = 0$ using engineering judgement. This nominal parameter set was used in simulations that reasonably predicted the closure rates of Room D, G, and Q closure rates (see Appendix B), which lends some credence to these nominal values. Nevertheless, three different parameter sets were utilized in the simulations herein: frictionless, nominal, and glued. The container compaction simulations only utilized nominal clay seams, while the empty room simulations were performed with all three clay seam parameter sets.

Table 2-2. Intended Clay Seam Parameter Sets

Parameter	Units	Frictionless	Nominal	Glued
μ_c	unitless	0	0.2	∞
σ_c	MPa	0	0	∞

Table 2-3. Actual Clay Seam Parameter Sets Used in Disposal Room Simulations

Parameter	Units	Frictionless	Nominal	Glued
$\mu_c^F = \mu_c^G$	unitless	0.5	0.5	∞
μ_c^{other}	unitless	0	0.2	∞
σ_c	MPa	0	0	∞

The intended clay seam parameter sets are listed in Table 2-2, while the clay seam parameter sets actually used in the disposal room simulations, including the CCO compaction simulations, are listed in Table 2-3. The nominal parameter set in Table 2-2 is consistent with the validation simulations in Appendix B. All simulations were supposed to use the same frictionless, nominal, and glued parameter sets, but it was discovered that, after Reedlunn et al. (2019) and Reedlunn and Bean (2019) were released, the Clay F and Clay G friction coefficients were inadvertently set to $\mu_c^F = \mu_c^G = 0.5$ instead of the intended value of 0.2 in the frictionless and nominal parameter sets used in the disposal room simulations. (See Fig. 2-1 for the locations of Clay F and G.) All friction coefficients for the other clay seams (μ_c^{other}) were set as intended in the disposal room simulations. The impact of this mistake on the final compacted container configuration was found to be relatively small (Reedlunn and Bean 2020b; Reedlunn and Bean 2021), so the original conclusions in Reedlunn et al. (2019) and Reedlunn and Bean (2019) remain unchanged. The mistake is noted herein simply to properly document the different simulation setups.

Note that Munson et al. (1989) only allowed the nine clay seams nearest Room D, labeled D through L to slide and/or separate. Presumably, their preliminary studies found that seams A, B, and M did not slide enough to be worth the computational expense. Herein, all twelve clay seams shown in Fig. 2-1 were allowed to slide and/or separate when the frictionless and nominal clay seam parameter sets were used. The disposal room is at a lower elevation than Room D and it has different dimensions, causing it to activate different seams than Room D. Without performing a separate study, it was prudent to let the mechanics determine which seams slide and/or separate, rather than artificially fixing seams.

2.1.4.2. Salt-to-Salt Sliding

Once the surfaces of the room came into contact with each other, the sliding was modeled using Coulomb friction. Although salt surfaces with a little water can bond together (heal) when held in contact for sufficient time and pressure, the Coulomb friction model did not include this behavior.

Stone (1997a), Park and Hansen (2005), and Park and Holland (2007) do not mention the coefficient of friction μ_s they utilized for salt-to-salt sliding, and Munson et al. (1989) did not simulate enough closure to cause salt-to-salt sliding. The SANTOS input files in the appendix of Stone (1997a), Park and Hansen (2005), and Park and Holland (2007) along with the SANTOS user guide (Stone 1997b), however, make it clear they used $\mu_s = 0$. Frictionless salt-to-salt contact seemed unrealistic to the authors, because salt-to-salt friction should be higher than clay seam friction ($\mu_s > \mu_c$). In addition, a large value of μ_s was desired to roughly approximate the effects of interface bonding. Given these considerations, the salt-to-salt coefficient of friction was selected to be $\mu_s = 0.5$.

2.2. Container Model

This section describes how the containers inside the disposal room were modeled.

2.2.1. Geometry and Boundary Conditions

Fig. 2-3 depicts the idealized container geometry with a $D_c = 0.574$ m diameter and $L_c = 0.870$ m height. One can compare this image with the schematic of the actual geometry in Figs. 1-4a and C-1 to see that the idealized geometry did not include details such as the ribs in the 55-gallon drum or the bolts in the flanged pipe. These features were thought to be insignificant compared to capturing aspects such as the pipe D_p/B_p and L_p/D_p ratios. See the Cubit journal file in Appendix D.2.2 for precise dimensions of the idealized geometry.

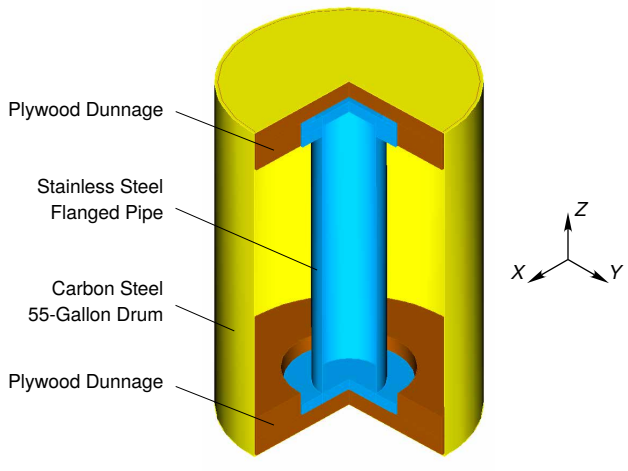
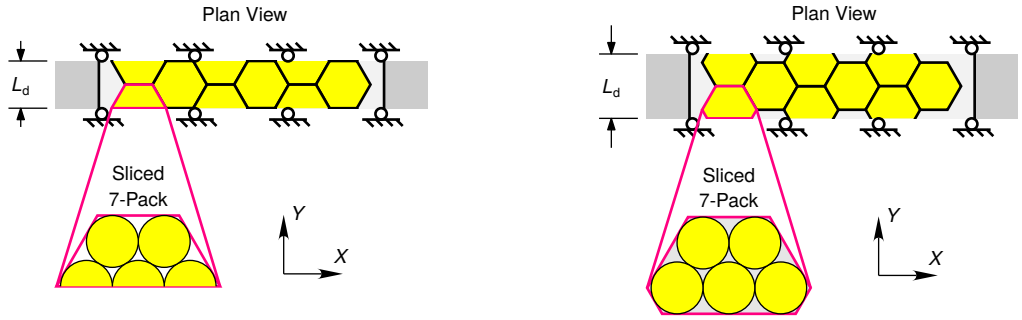


Figure 2-3. Idealized container geometry. (Quarter wedge removed for visualization purposes.)

The containers were placed in the room with only a small 5 mm gap between their D_c diameters and they were perfectly stacked. The 5 mm spacing between the container drums conservatively ignored the drum ribs and bolt ring that might have made the effective diameter as large as 0.609 m. See the Cubit journal file in Appendix D.2.3 for precise emplacement dimensions. In reality, emplaced containers will have slight deviations from this ideal arrangement, but such

geometric deviations were not considered. The deviations caused by imprecise emplacement were assumed to be insignificant compared to container movement as the roof and floor become curved during room closure.

Representing each container explicitly, for the reasons discussed in Chapter 1, made it computationally prohibitive to include all the containers in an entire disposal room. If a 91.44 m long disposal room was filled with CCO containers, it would have 7182 containers. Instead, the many rows in Fig. 1-2a were initially reduced to the depth of one 7-pack (L_d = 1.58 m) with rollers enforcing zero normal displacement at the Y-boundaries of the domain, as shown in Fig. 2-4a. The Y-boundaries sliced some containers in half and left others intact, so zero Y displacement at the Y-boundaries was prescribed at the surfaces exposed by the slice, and contact with an analytic plane kept the intact containers from penetrating the Y-boundaries. Unfortunately, once half-containers contacted the left and right salt walls, the half-containers would often penetrate the walls. These anomalous penetrations were likely due to a conflict between the contact constraint and the zero Y displacement. Further study might be able to resolve this issue, but an adjusted container configuration was utilized for the time being. As shown in Fig. 2-4b, the Y-boundaries were moved outward by a container radius on each side (L_d = 2.15 m) and the remaining half-containers were simply eliminated. These undesirable gaps along the Y-boundaries may have an impact on the container compaction predictions, but the impact was assumed small enough to not be considered in the current study.



(a) Attempted container configuration **(b) Adapted container configuration**

Figure 2-4. (a) Attempted container configuration that exhibited penetration issues, and (b) adapted container configuration that was utilized in the compaction simulations.

2.2.2. Material Models

This subsection describes the material models and parameters used to represent the mechanical behavior of the carbon steel, stainless steel, and plywood in the containers. The waste itself is not included in the container model because no requirement exists for how much waste must be placed in a given container. Neglecting the structural resistance and mass contributions of the waste was deemed a conservative assumption.

Sacks full of magnesium oxide (MgO) are placed on top of a portion of the container stacks at the WIPP, but they also were not considered in this work. A numerical method to include the MgO

sacks has not been established to the authors' knowledge and several past studies of room closure have ignored the sacks (Stone 1997a; Park and Hansen 2005; Park and Holland 2007). In reality, once the roof applies a small amount of pressure, the sacks will likely break open and spill their contents between the containers. Consequently, the sacks are expected to supply an insignificant amount of structural resistance to closure until the waste becomes nearly fully compacted. Upon full compaction, the MgO granules are expected to occupy the space between the containers, which would only increase the spacing between containers, and thus it was considered conservative to neglect the granules.

The models for the carbon steel, stainless steel, and plywood were chosen to be relatively simple while capturing what were thought to be the most relevant physical behaviors. At the time of this writing, laboratory tests to characterize the behavior of the CCO materials have only been performed on the carbon steel used to make the 55-gallon drum. Consequently, material models and parameters for the stainless steel and plywood were selected from published literature.

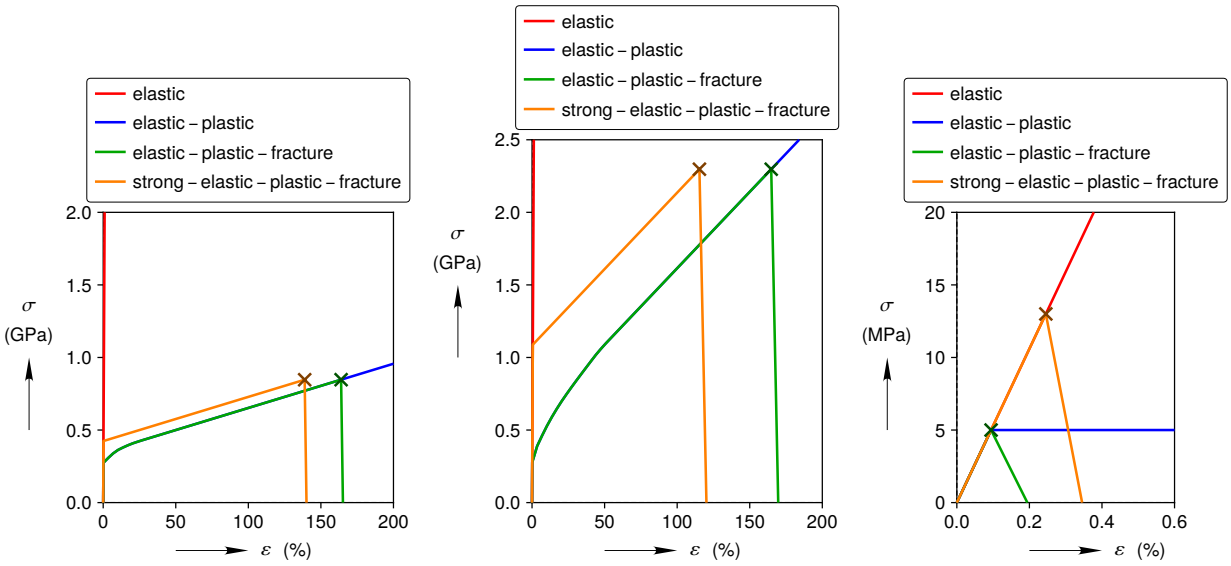
The stainless steel and plywood material models employed in Park and Hansen (2004) were initially considered for use here since POCs are relatively similar to CCO containers. Park and Hansen (2004) obtained their material models from Ludwigsen et al. (1998), except Park and Hansen (2004) allowed the 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. The values they chose appear to be reasonable values, and one could probably justify them by comparing the mechanical responses produced by those parameters against typical, experimentally measured, mechanical responses in published literature. On the other hand, fracture models for the stainless steel and plywood were also desired for this work, and Ludwigsen et al. (1998) did not include fracture. Therefore, experimental measurements were found in the literature and new material model calibrations, including fracture, were developed for the stainless steel and plywood.

Each CCO material was represented with four different material models:

1. elastic
2. elastic-plastic
3. elastic-plastic-fracture
4. strong-elastic-plastic-failure.

(The elastic-plastic-fracture and strong-elastic-plastic-failure models for the plywood do not include plasticity, but these models are still referred to as elastic-plastic-fracture and strong-elastic-plastic-failure for simplicity.) The details of the models are discussed in the following three sub-sections. For now, a summary of the true stress σ vs. true strain ε behavior of each model under uniaxial tension is shown in Fig. 2-5. Material failure is marked with a $\times$, after which the stress linearly decays to zero. Note that the elastic-plastic model for a given material utilizes the same elastic properties as the corresponding elastic model, and the elastic-plastic-fracture model utilizes the same elastic and plastic properties as the elastic-plastic model. Each strong-elastic-plastic-fracture model was developed to have a yield strength in between the elastic-plastic-fracture model's yield strength and the elastic model's infinite

strength. Thus, the first three models form a hierarchy, while the fourth model does not quite fit into that hierarchy.



(a) Carbon Steel **(b) Stainless Steel** **(c) Plywood**

Figure 2-5. True stress vs. true strain curves under uniaxial tension for container materials

The container behavior was varied for the following three reasons:

1. Even if the material properties were all mechanically tested to determine material parameters more precisely for an individual specimen, the specimen would represent material properties from a specific heat/batch of material and not necessarily represent the lowest or highest allowable values. Thus, significant variation in material properties is allowed and should be considered.

2. The material models in the next three sub-subsections are all rate-independent models. Although viscous effects in metals loaded at room temperature and slow strain rates are typically ignored in engineering analyses, they are present (Deibler 2014). The surrounding rock mass will apply loads to the containers over far longer times than considered in typical engineering analyses. This means a significant portion of the container structural resistance may relax before the closing room can apply further compaction.

3. The material models do not include degradation. If water or brine is transported into the disposal room along either clay seams or cracks in the anhydrite layers, the plywood will rot and the metals will corrode. Carbon steel is susceptible to corrosion, but austenitic stainless steels, such as 304L, are also susceptible to corrosion when they are under stress (stress corrosion cracking). Thus, one further case was added to the material model list: complete degradation of container materials to zero stiffness. This case was captured in Chapter 3 by simulating an empty room without any containers. Note that zero stiffness containers will, of course, compact to zero volume and become super-critical, so this case should be seen as a lower bound for compaction rather than representing a potential

situation. A realistic criticality analysis of degraded canisters would still need to assign some spacing to the nuclear waste.

Varying the container material models should bound the uncertainties in the container mechanical behavior. Most likely, the actual container behavior will be somewhere between the elastic-plastic-fracture model and the empty room model.

2.2.2.1. Carbon Steel

The CCO drawing in Fig. C-1 does not specify a material for the 55-gallon drum other than carbon steel coated with a rust inhibitor. The drawing, however, does list part number CQ5508 for the 55-gallon drum and suggests Skolnik Industries as a supplier. A drawing of part number CQ5508 was found on Skolnik Industries website (Skolnik Industries 2018), and the drawing specifies "carbon steel per ASTM A1008 and ASTM 568 or equivalent". (ASTM A1008 is a more specific material standard under the umbrella of the ASTM568 standard.)

The ASTM A1008 carbon steel was modeled using the `ml_ep_fail` model in Sierra/SolidMechanics (2018). This model is an isotropic, hypoelastic, rate-independent, von Mises plasticity model with the tearing parameter failure model (Wellman 2012). The model uses the strain rate decomposition in Eq. (2.2) and the linear elastic relationship given by Eqs. (2.3) and (2.4). The plastic strain evolution is governed by an associated flow rule,

$$\dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}} = \dot{\bar{\varepsilon}}^{\mathrm{pl}} \frac{\partial f}{\partial \boldsymbol{\sigma}}, \tag{2.10}$$

where $\dot{\bar{\varepsilon}}^{\mathrm{pl}}$ is the rate of the equivalent plastic strain and f is the yield function, which is constrained to be $f \leq 0$. The equivalent plastic strain only evolves if the yield function is satisfied ($f = 0$) and the satisfaction persists ($\dot{f} = 0$). Otherwise, $\dot{\bar{\varepsilon}}^{\mathrm{pl}} = 0$. The yield function is defined as

$$f = \bar{\sigma} - H(\bar{\varepsilon}^{\mathrm{pl}}), \tag{2.11}$$

where $\bar{\sigma}$ is the equivalent stress and $H(\bar{\varepsilon}^{\mathrm{pl}})$ is a user specified isotropic hardening curve. The user can specify the relative fraction of isotropic vs. kinematic hardening by setting $0 \leq \beta \leq 1$, where $\beta = 0$ is kinematic and $\beta = 1$ is isotropic. Here, β was set to 1, in which case the equivalent stress $\bar{\sigma} = \sqrt{3 J_2}$. The tearing parameter ξ is calculated as

$$\xi = \int_0^{\bar{\varepsilon}^{\mathrm{pl}}} \left\langle \frac{2\,\sigma_1}{3\,(\sigma_1 - \sigma_{\mathrm{m}})} \right\rangle^n \mathrm{d}\hat{\bar{\varepsilon}}^{\mathrm{pl}} \tag{2.12}$$

where σ_1 is the maximum principal stress (tensile stresses are positive), $\sigma_{\mathrm{m}} = I_1/3$ is the mean stress, n is a material parameter and $< \cdot >$ are Macaulay brackets. Failure initiates when $\xi = \xi_{\mathrm{f}}$, where ξ_{f} is a material parameter. Upon failure initiation, the direction of the maximum principal stress is stored and subsequent strain along that direction $\varepsilon_{\mathrm{cos}}$ is monitored as the "crack opening strain". The stress components are linearly decayed to zero as $\varepsilon_{\mathrm{cos}}$ grows from zero to $\varepsilon_{\mathrm{ccos}}$, which is a material parameter called the critical crack opening strain.

Young's modulus of elasticity E, Poisson's ratio ν, and the mass density ρ were taken from ASME (2015) for ASTM A1008 carbon steel. The values are listed in Table 2-6. These were

Table 2-4. Carbon steel hardening curve data points

$\bar{\varepsilon}^{\mathrm{pl}}$ (unitless)	H (MPa)
0.000	274.0
0.015	276.5
0.025	295.6
0.035	310.4
0.045	322.0
0.050	327.0
0.100	362.5
0.150	387.5
0.200	407.7
0.250	424.4
1.634	846.6

(a) Nominal

$\bar{\varepsilon}^{\mathrm{pl}}$ (unitless)	H (MPa)
0.000	424.4
1.384	846.6

(b) Strong

converted to bulk modulus B and shear modulus μ using the standard isotropic elastic stiffness formulas: $B = E/(3\,(1-2\,\nu))$ and $\mu = E/(2\,(1+\nu))$.

The nominal hardening curve for the elastic-plastic and elastic-plastic-fracture models was calibrated against uniaxial tension tests on specimens cut from a 55-gallon drum lid taken to be representative of the drum material to be used in CCOs. See Appendix C.2 for details of the experiments and the procedure used to determine the hardening curve $H(\bar{\varepsilon}^{\mathrm{pl}})$. In short, the engineering stress-strain curve was converted to a true stress-strain curve by assuming isochoric plastic flow between initial yield and the ultimate tensile (engineering) stress $(F/A)_{\mathrm{uts}}$, and by assuming isochoric plastic flow plus linear hardening between $(F/A)_{\mathrm{uts}}$ and the fracture stress σ_{f}. The `ml_ep_fail` model accepts a series of points on the hardening curve and linearly interpolates between them. The points are listed in Table 2-5a and the curve is plotted as the 'nominal' curve in Fig. 2-6.

The strong hardening curve for the strong-elastic-plastic-fracture model was created by treating the ASTM A1008 carbon steel as if it had been previously cold worked up to $(F/A)_{\mathrm{uts}}$ in a uniaxial tensile test. In other words, the 'strong' curve in Fig. 2-6 was created by deleting the solid portion of the nominal curve and shifting the remaining dashed nominal curve over to the H axis. Starting the strong hardening curve at the true stress corresponding to the ultimate tensile (engineering) stress is admittedly somewhat arbitrary. A maximum load in a tensile test just indicates the onset of a necking instability, and one can reach higher stresses without necking if one applies, say, biaxial tension (Chen et al. 2018). On the other hand, this method creates a material that is far stronger than typically found in the published literature for ASTM A1008 carbon steel, without creating something physically inconceivable.

Ductile failure models and numerical methods to handle failure are ongoing subjects of research without consensus on the proper approach for a given situation. Therefore, rather than expend a large degree of effort calibrating the ideal model, the tearing parameter model was calibrated in an expedient manner. If a particular aspect of the container compaction simulations turns out to be sensitive to failure, the model selection and calibration can be revisited in the future.

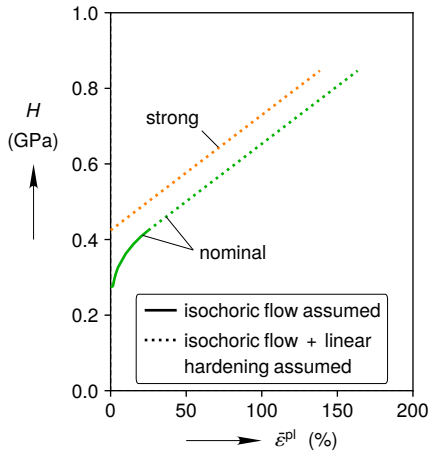


Figure 2-6. Carbon steel hardening curves

The tearing parameter at fracture ξ_f is relatively easy to calibrate. In uniaxial tension, Eq. (2.12) reduces to

$$\xi = \int_0^{\bar{\varepsilon}^{pl}} d\hat{\bar{\varepsilon}}^{pl} = \bar{\varepsilon}^{pl}. \tag{2.13}$$

Thus, ξ_f is simply the final equivalent plastic strain in the hardening curve $\bar{\varepsilon}_f^{pl}$.

Although uniaxial failure is independent of the exponent n, the exponent n has a significant impact on how the failure model behaves under multi-axial states of stress, as shown in Section 4.5 of Corona and Reedlunn (2013). In the absence of experimental data to calibrate n against, other than uniaxial tension, the default value of $n = 4$ was adopted here.

The purpose of a non-zero critical crack opening strain ε_{ccos} is to dissipate the energy associated with generating fracture surfaces. The amount of energy dissipated, however, will depend on the element size because the stress is decayed to zero over the whole element. Wellman (2012) attempted to find the proper relationship between ε_{ccos} and element size such that energy dissipated remained mesh independent, but it was not terribly successful. Here engineering judgement was used to select $\varepsilon_{ccos} = 0.013$.

Parameters for each of the carbon steel material model calibrations are listed in Table 2-6.

2.2.2.2. Stainless Steel

The CCO drawing in Fig. C-1 specifies that the flanged pipe will be made of ASTM A182, grade F304/F304L, stainless steel for the flanges and ASTM A312, grade TP304/TP304L, stainless steel for the pipe. Both ASTM A182 and A312 are different forms of 304 or 304L stainless steel. For simplicity, only 304L stainless steel will be considered here.

The 304L stainless steel was modeled using the same general `ml_ep_fail` model with isotropic hardening ($\beta = 1$) utilized for the carbon steel. The parameter values, however, differed from the carbon steel.

Table 2-6. Carbon Steel Calibrations

Parameter	Units	Elastic	Elastic-Plastic	Elastic-Plastic-Fracture	Strong-Elastic-Plastic-Fracture
E	GPa	202.0	202.0	202.0	202.0
ν	unitless	0.3	0.3	0.3	0.3
ρ	kg/m^3	7750.0	7750.0	7750.0	7750.0
β	unitless	-	1.0	1.0	1.0
$H(\bar{\varepsilon}^{\text{pl}})$	-	-	Table 2-5a	Table 2-5a	Table 2-5b
ξ_{f}	unitless	-	-	1.634	1.384
n	unitless	-	-	4	4
$\varepsilon_{\text{ccos}}$	unitless	-	-	0.013	0.013

The elastic constants E and ν, and the density ρ, were taken from Table 2.7.1.0b3 of the Metallic Materials Properties Development and Standardization (MMPDS) Handbook (William J. Hughes Technical Center et al. 2015) for solution heat treated AISI 304 stainless steel[†]. The values are listed in Table 2-9.

Table 2-7. Stainless steel hardening curve data points

$\bar{\varepsilon}^{\text{pl}}$ (unitless)	H (MPa)
0.0	277.2
0.0047	307.7
0.0109	331.4
0.0232	366.1
0.0479	422.7
0.0974	523.3
0.1470	613.8
0.1966	692.4
0.2462	765.1
0.2959	833.6
0.3456	900.4
0.3952	966.4
0.4449	1031
0.4946	1088
1.637	2297

(a) Nominal

$\bar{\varepsilon}^{\text{pl}}$ (unitless)	H (MPa)
0.0	1088
1.142	2297

(b) Strong

The hardening curve in the elastic-plastic and elastic-plastic-fracture models was calibrated against the "RT" tensile response of 304L base material in Fig. 4 of Blandford et al. (2007). Although Blandford et al. (2007) does not make it explicitly clear, the engineering stress-strain RT curve in their Fig. 4 corresponds to the 21.1 °C (70°F) for heat 72K9 in Table 2 and the true

[†]Table 2.7.1.0b3 of William J. Hughes Technical Center et al. (2015) lists 199.9 GPa and 193.0 GPa for Young's modulus in tension and compression, respectively. These two values were averaged and then inadvertently scaled by 1.03. This 3 % error is noted only for completeness, since it should have negligible impact on the quantities of interest herein.

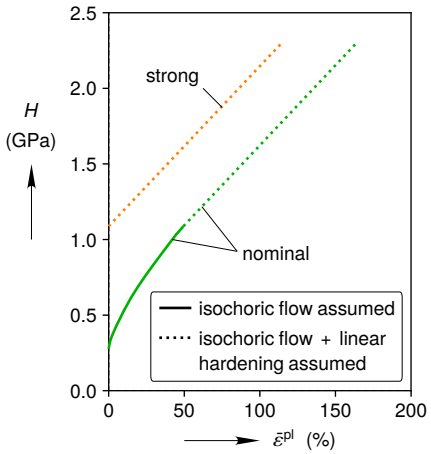


Figure 2-7. Stainless steel hardening curve

stress-strain RT curve in their Fig. 5. A few simple calculations confirmed that they used the procedure described in Appendix C.2.3 to create their true stress-strain RT curve in Fig. 5. This curve was digitized and the plastic strain $\bar{\varepsilon}^{pl}$ was computed as $\varepsilon - \sigma/E$ to create the hardening curve. The points are listed in Table 2-7 and they are plotted as the 'nominal' curve in Fig. 2-7.

Hardening for the strong-elastic-plastic-fracture model was computed in the same manner as the carbon steel strong-elastic-plastic-fracture model. Austenitic stainless steels are strengthened principally via strain hardening, so this approach to creating the strong hardening curve is particularly suitable. See Table 2-8b and Fig. 2-7 for the data points and a plot of the 'strong' curve.

Fracture in the elastic-plastic-fracture model was also calibrated in the same way as the carbon steel failure model. The tearing parameter at failure ξ_f was the final equivalent plastic strain in the hardening curve $\bar{\varepsilon}_f^{pl}$, $n = 4$ was assumed, and $\varepsilon_{ccos} = 0.048$ was selected based on experience.

Parameters for each of the stainless steel material model calibrations are listed in Table 2-9.

Table 2-9. Stainless Steel Calibrations

Parameter	Units	Elastic	Elastic-Plastic	Elastic-Plastic-Fracture	Strong-Elastic-Plastic-Fracture
E	GPa	202.5[†]	202.5	202.5	202.5
ν	unitless	0.27	0.27	0.27	0.27
ρ	kg/m^3	7916.5	7916.5	7916.5	7916.5
β	unitless	-	1.0	1.0	1.0
$H(\bar{\varepsilon}^{pl})$	-	-	Table 2-8a	Table 2-8a	Table 2-8b
ξ_f	unitless	-	-	1.637	1.142
n	unitless	-	-	4	4
ε_{ccos}	unitless	-	-	0.048	0.048

2.2.2.3. Plywood

According to Fig. C-1, the plywood dunnage in the CCO is only required to be APA Exterior Grade C-C (Standards and Technology 2010). The exterior rating means waterproof glues must be used to bond the plies together, while the C-C grade applies to the visual appearance of the veneers (see Section 5.4.4 of Standards and Technology (2010)). These ratings do not specify the type of wood for each ply, the ply thickness, or the number of plies.

Pine and fir are probably the most common types of exterior plywood. Pine is weaker, cheaper, and readily available in the southeast United States so this work assumes the plywood is made from Pine plies.

Wood is orthotropic with stiffness along the grain being roughly 20× stiffer than transverse to the grain (see Table 5-1 in Ross (2010)). The effective anisotropy of plywood is much less than plain wood because the grain of each ply is 90 degrees to the neighboring plies. Although the number of plies per sheet of plywood has not been specified, Fig. C-2 shows five sheets of plywood attached with wood screws. The grains of each sheet will probably be randomly oriented, within the plane of the sheet, to the grains of the other four sheets, such that one could treat the five sheets as a monolithic transversely isotropic material. On the other hand, the plywood will most likely be loaded in bending, where the in-plane effective stiffness and strength will dominate the plywood's response. Thus, the plywood assembly was modeled as an isotropic material, using the in-plane properties as isotropic properties.

The plywood elastic-plastic model included plastic deformation, but not fracture, so the `multilinear_ep` model in Sierra/SolidMechanics (2018) was utilized. This model is identical to `ml_ep_fail`, except it does not have a fracture model. Nevertheless, the material was assigned a flat hardening curve with $H = \sigma_{\mathrm{f}}$, which restrained the von Mises stress to be no larger than σ_{f} a user specified "failure" strength. Perfect-plasticity models such as this are often used to model diffuse failure where the material crumbles, but can still sustain some stress.

The `elastic_fracture` model in Sierra/SolidMechanics (2018) was used for the elastic-plastic-fracture and strong-elastic-plastic-fracture models. The `elastic_fracture` model also utilizes the isotropic, hypoelastic, linear elastic relationship given by Eqs. (2.3) and (2.4), but it does not include plasticity. Once the maximum tensile principal stress σ_1 reaches the user specified maximum allowable value of σ_{f}, failure initiates. Similar to the tearing parameter failure model inside `ml_ep_fail`, the stress components are linearly decayed to zero as $\varepsilon_{\mathrm{cos}}$ grows from zero to $\varepsilon_{\mathrm{ccos}}$.

Table 2-10. Typical stiffness and strength measurements on pine plywood

Parameter	Units	McNatt et al. (1990)	Biblis (2000)	Wilczyński and Warmbier (2012)
ρ	kg/m^3	577	-	480
$E^{\parallel}$	GPa	9.3	7.7	13.79
$E^{\perp}$	GPa	1.2	-	1.05
$\sigma_{\mathrm{f}}^{\parallel}$	MPa	29	37	-
$\sigma_{\mathrm{f}}^{\perp}$	MPa	20	-	-

The stiffness and strength of plywood depend on the type of wood, ply thickness, number of plies, moisture content, defect distribution, and a myriad of other factors. The current authors were unable to find a definitive reference for plywood properties, so the measurements of three different studies on pine plywood are shown in Table 2-10. The stiffness along the grain $E^{\parallel}$ and perpendicular to the grain $E^{\perp}$ are listed along with the strength (modulus of rupture) along the grain $\sigma_{\mathrm{f}}^{\parallel}$ and perpendicular to the grain $\sigma_{\mathrm{f}}^{\perp}$. The moisture content, number of plies, and ply thickness were not identical in these studies, so the measurements are not directly comparable, but they do give a rough idea of the variability. McNatt et al. (1990) measured stiffness and strength, parallel and perpendicular to the grain, while the other studies only measured some of these properties, so the McNatt et al. (1990) measurements were used herein. The McNatt et al. (1990) values listed in Table 2-10 were measured using the pure moment method on 3-ply, 9.5 mm thick, southern pine, plywood with 10.5 % moisture content.

Table 2-11. Plywood Calibrations

Parameter	Units	Elastic	Elastic-Plastic	Elastic-Plastic-Fracture	Strong-Elastic-Plastic-Fracture
E	GPa	5.3	5.3	5.3	5.3
ν	unitless	0.35	0.35	0.35	0.35
ρ	kg/m^3	577	577	577	577
β	unitless	-	1.0	-	-
$H(\bar{\varepsilon}^{\mathrm{pl}})$	-	-	No Hardening	-	-
σ_{f}	MPa	-	-	5	13
$\varepsilon_{\mathrm{ccos}}$	unitless	-	-	0.001	0.001

Given the uncertainties, simple methods were used to select the elastic properties instead of using composite panel theory. The isotropic Young's modulus was calculated as a simple average of $E^{\parallel}$ and $E^{\perp}$ to obtain $E = 5.3$ GPa. McNatt et al. (1990) did not measure Poisson's ratio, so $\nu = 0.35$ was selected after qualitatively considering the orthotropic Poisson's ratios for Loblolly Pine (a common southern pine) listed in Table 5-2 of Ross (2010). McNatt et al. (1990) also measured the mass density as $\rho = 577$ kg/m^3.

Wood failure is known to depend on the duration of applied bending stress (Holzer et al. 2007). McNatt et al. (1990) did not specify the duration of their tests, but they were likely short-term tests that lasted on the order of 10 minutes. Hoffmeyer and Sørensen (2007) conducted constant bending stress tests under laboratory controlled conditions for as long as 13 years, which are the longest tests known to the authors. Their results were reported in terms of f/f_0, where f was the applied bending stress and f_0 was the short term strength. At 20 % moisture content, the longest lasting sample survived for $\approx$ 10 yr at $f/f_0 = 53$ %. They did not detect a threshold stress beneath which the wood remains intact indefinitely: the data in the semi-log plot in Fig. 6 of Hoffmeyer and Sørensen (2007) looks quite linear, without any sign of flattening out for longer stress durations. Wooden structures can obviously survive for hundreds of years, so there must be a threshold stress, even if it takes excessively long to measure. This threshold stress must be between 0 % and 53 % of the short term strength.

A sophisticated model of wood failure would involve a damage variable computed as an integral

over the stress history, but the model here is quite simple. Consequently, the failure stress was taken to be the threshold stress. In the face of the uncertainty in the threshold stress, $f/f_0 = 20$ % and 53 % were selected for the elastic-plastic-failure and strong-elastic-plastic-failure models, respectively. After computing the isotropic short-term strength as (29+20)/2=24.5 MPa, the elastic-plastic-fracture and strong-elastic-plastic-fracture failure strengths were respectively computed as $\sigma_f = 0.2 \times 24.5 \approx 5$ MPa and $\sigma_f = 0.53 \times 24.5 \approx 13$ MPa. As before, the critical crack opening strain of $\varepsilon_{ccos} = 0.001$ was selected using engineering judgement.

A summary of the plywood material parameters is shown in Table 2-11.

2.2.3. Interface Models

2.2.3.1. Salt-to-Metal/Plywood Sliding

Coulomb friction was used to model the sliding contact between salt and metal, and between salt and plywood with a μ_{smp} salt-to-metal/plywood coefficient of friction. Stone (1997a), Park and Hansen (2005), and Park and Holland (2007) assumed $\mu_{smp} = 0$, which again seems too small to the current authors. The salt-to-metal coefficient of friction is likely somewhere between $\mu_c = 0.2$ and $\mu_s = 0.5$, so $\mu_{smp} = 0.4$ was selected.

2.2.3.2. Metal/Plywood-to-Metal/Plywood Sliding

Once again, Coulomb friction was used to model the sliding contact between the metal surfaces, between metal and plywood surfaces, and between plywood surfaces with a μ_{mp} metal/plywood-to-metal/plywood coefficient of friction. One could perhaps specify independent coefficients of friction for each of these pairs of contact surfaces, but a single coefficient of fraction of $\mu_{mp} = 0.3$ was utilized herein for simplicity.

2.3. Numerical Approach

This section describes the numerical approaches considered (Section 2.3.1) and the approach eventually selected (Section 2.3.2). Appendix A goes into further detail on how numerical settings, such as element size, number of contact iterations, and density scaling, were selected.

2.3.1. Potential Approaches

Several different approaches were considered to run the container compaction model described in Sections 2.1 and 2.2. Sierra/SolidMechanics (2018) has three standard approaches for simulating mechanical events, each with advantages and disadvantages:

1. Explicit dynamics

   Explicit dynamics is quite robust at handling contact, severe deformations, and element death, but numerical stability requires a very small time step, typically on the order of 1 μs. This makes explicit dynamics well suited to simulating short duration events, such as car crashes, but anything that lasts longer than a few seconds is not practical.

2. Implicit quasi-statics

   Implicit quasi-statics can handle arbitrarily large time steps if the solver can converge, but contact chatter, intense deformations, and element death often hamper solution convergence.

3. Implicit dynamics

   Implicit dynamics can also accommodate arbitrarily large time steps when the solver converges, and the addition of inertia can sometimes improve convergence over implicit quasi-statics. Inertia penalizes large local displacements that often occur as contact surfaces chatter and when an element dies.

Initial attempts to simulate the container compaction with implicit quasi-statics and implicit dynamics were plagued with solver convergence issues from the onset of the simulation. After simulating only a few seconds of room closure the implicit quasi-static solver would either fail to converge or would invert elements in the container-to-salt contact regions. The implicit dynamics simulations would sometimes run a few seconds further, but one or more containers would inevitably jump upwards. These strange jumping containers would knock other containers over and cause the time step size to plummet to prohibitively small values. Despite several attempts to troubleshoot the issue, a root cause was not found. Implicit solver convergence issues were expected to only worsen once the containers were severely compressed, so the fully implicit approach was abandoned.

Two “innovative” approaches were subsequently considered. In the first approach, the geomechanical model with implicit quasi-statics was used to predict the closure of an empty room. The displacement of the empty room’s surfaces was then used as time varying essential boundary conditions in a separate simulation of container compaction. After uncoupling the container compaction from the geomechanics, the container compaction was sped up enough to use explicit dynamics. This uncoupled approach, of course, neglected any impact the containers had on the shape of the closing room and made it difficult to determine when the structural resistance of the containers had balanced with the crushing force of the salt. Unless a termination condition was applied, the room would apply whatever forces were necessary to compact the containers to zero volume. The second approach, which was ultimately utilized herein, sped up the salt viscoplastic strain rate by several orders of magnitude so that explicit dynamics could be used in a single, coupled, simulation of container compaction due to room closure.

2.3.2. Viscoplastic Strain Rate Scaling

The notion of scaling the visocoplastic strain rate can be made more precise with a few definitions. Define the physical time as t, the momentum balance time utilized by the explicit

dynamics code as t^{mb}, and the equivalent viscoplastic strain as $\bar{\varepsilon}^{\text{vp}}$ (see Section 2.1 of Reedlunn (2018a) for a full definition of $\bar{\varepsilon}^{\text{vp}}$). The viscoplastic scale factor $s = s(t^{\text{mb}})$ is defined as

$$s = \frac{\dfrac{\mathrm{d}\bar{\varepsilon}^{\text{vp}}}{\mathrm{d}t^{\text{mb}}}}{\dfrac{\mathrm{d}\bar{\varepsilon}^{\text{vp}}}{\mathrm{d}t}} = \frac{\mathrm{d}t}{\mathrm{d}t^{\text{mb}}}, \tag{2.14}$$

so one can calculate the current time from the scale factor history as

$$t = \int_{t_{\text{i}}^{\text{mb}}}^{t^{\text{mb}}} s \, \mathrm{d}\hat{t}^{\text{mb}} + t_{\text{i}}, \tag{2.15}$$

where t_{i} and t_{i}^{mb} respectively are the physical and momentum balance time at the start of the time period of interest.

The viscoplastic rate scaling had to be fast enough to reach the final container configuration in a reasonable amount of run time, but not so fast that inertial effects significantly altered the predictions. The initial plan to scale the viscoplastic strain rate was to simply multiply several material parameters by a single constant scale factor s. Converting between t and t^{mb} would have been simply $t = s\,t^{\text{mb}}$. Unfortunately, a small value of s was required to avoid inertial effects when the room closed rapidly immediately after excavation. Once the closure rate slowed down, the small value of s resulted in excessively long simulation run times. Instead, inertial effects and long run times were sidestepped by starting with a small value of s and increasing it over the course of the simulation.

A few different approaches were considered to vary s. One approach would have been to start the simulation with one set of material parameters corresponding to a small s value, run past the rapid room closure period, stop the simulation, and then restart with a new set of material parameters corresponding to a large s value. Besides being awkward, this method is probably fraught with issues since Sierra/Solid Mechanics's restart capability was not built to handle constitutive model parameter changes. Another approach was utilized in Reedlunn (2017): the temperature T was varied to continuously vary $\dot{\bar{\varepsilon}}^{\text{vp}}$ over time. Only isothermal simulations were considered in that work, so it was possible to freely vary T after a few modifications to the MD model parameters. This work utilizes a similar, yet slightly cleaner, approach. Sierra/SolidMechanics (2018) allows users to modify material model internal variables in the middle of the simulation, so the MD model was modified to include s as an internal variable inside the `md_viscoplastic` implementation (see Section 3.3 of Reedlunn (2018a)). A s vs. t^{mb} profile was specified a-priori, the value of s was read into the `md_viscoplastic` implementation, and the physical time t was calculated using Eq. (2.15). See Appendices D.3.1 and D.3.2 for examples of modifying s and calculating t in a Sierra/Solid Mechanics input file.

This viscoplastic rate scaling procedure was verified in Appendix A.2.1 by comparing explicit dynamic simulations of empty room closure against a reference implicit quasi-static simulation. Successively slower s vs. t^{mb} profiles were compared to find the profile that approximated the quasi-static solution, yet still ran relatively quickly. Although the profile chosen in Appendix A.2.1 agreed very well with the quasi-static solution, this profile was not used with

impunity in the container compaction simulations. The ratio of the kinetic energy over the internal energy E_k/E_i was monitored to ensure inertial effects were negligibly small once containers were added to the room. All simulations had an initial spike in E_k/E_i immediately following excavation, but this was shown to not have lasting effects in Appendix A.2.1. After the initial spike, the maximum value observed was $E_k/E_i = 1.1$ %.

Speeding up the salt viscoplasticity also has the effect of mixing up time scales. Finite viscoplastic deformation may occur over a dynamic event, such as the sudden local slipping of one surface against another, even though, in reality, the viscoplastic deformation should be virtually zero over a short duration event. This error should be negligible for the simulations herein. For example, at $t = 100$ yr, the vertical closure rate of an empty disposal room is predicted to be $\dot{\delta}_v/L_v = 1.98\times10^{-8}$ %/s (see the $L_h/192$ curve in Fig. A-7) and $s = 3.55\times10^{9}$ (see the $s_1 = 2\times10^{9}$ s^{-1} curves in Fig. A-3). If the dynamic event occurred over a relatively long $\Delta t^{mb} = 1$ ms, the vertical room closure over that duration would still only be $\delta_v/L_v = (1.98\times10^{-8}$ %/s$)(3.55\times10^{9})(10^{-3}$ s$) = 0.07$ %.

3. RESULTS

The compaction predictions in this chapter are broken into two sections. Section 3.1 discusses the sensitivity to container mechanical behavior, while Section 3.2 covers the sensitivity to clay seam geomechanical behavior.

3.1. Sensitivity to Container Behavior

Images of the container deformation for each container model are discussed in Sections 3.1.1 to 3.1.4, followed by quantitative comparisons in Section 3.1.5.

3.1.1. Elastic

The compaction of the elastic containers is shown on the left side of Fig. 3-1. Each column depicts a different canister model and each row corresponds to a snapshot in time since the room was excavated. The 55-gallon drums were included in the simulation, but they are not shown in Fig. 3-1 in order to reveal the plywood and stainless steel flanged pipe deformation. The plywood is colored brown, while the stainless steel flanged pipe is colored blue. The surrounding geology was also removed for visualization purposes, leaving only the (semi-transparent) surface of the room.

Room closure rearranges the elastic containers, but negligible container deformation occurs. The roof first contacts the container stacks in the center of the room at t = 35 yr, and all container stacks are in contact with the roof by 70 yr. The containers are under a significant amount of vertical load, yet the stainless steel pipes do not column buckle. As a result, the vertical closure remains propped open while the horizontal closure continues to evolve. This horizontal closure is accommodated by containers moving into a tighter close packed arrangement, plywood pieces shifting up/down enough to slide over/under one another, and containers rotating about their cylindrical axis to align the neighboring element facets for maximum compaction. Containers also move into the open gaps left by deleting the half drums (see Section 2.2.1), but those gaps primarily fill with salt deforming into the room, as shown by the semi-transparent room surface in Fig. 3-1.

The elastic containers are apparently too stiff to buckle, but the stresses are also unreasonably high. According to Table 2.7.1.0b3 of MMPDS Handbook, 304 stainless steel has a yield strength between 0.18 and 1.05 GPa. (In the authors' experience, yield stresses are typically closer to 0.18 GPa than 1.05 GPa.) A plot of the von Mises stress distribution in the stainless steel pipes in Fig. 3-2 shows that most locations have greater than 1.0 GPa of stress. Plastic deformation is

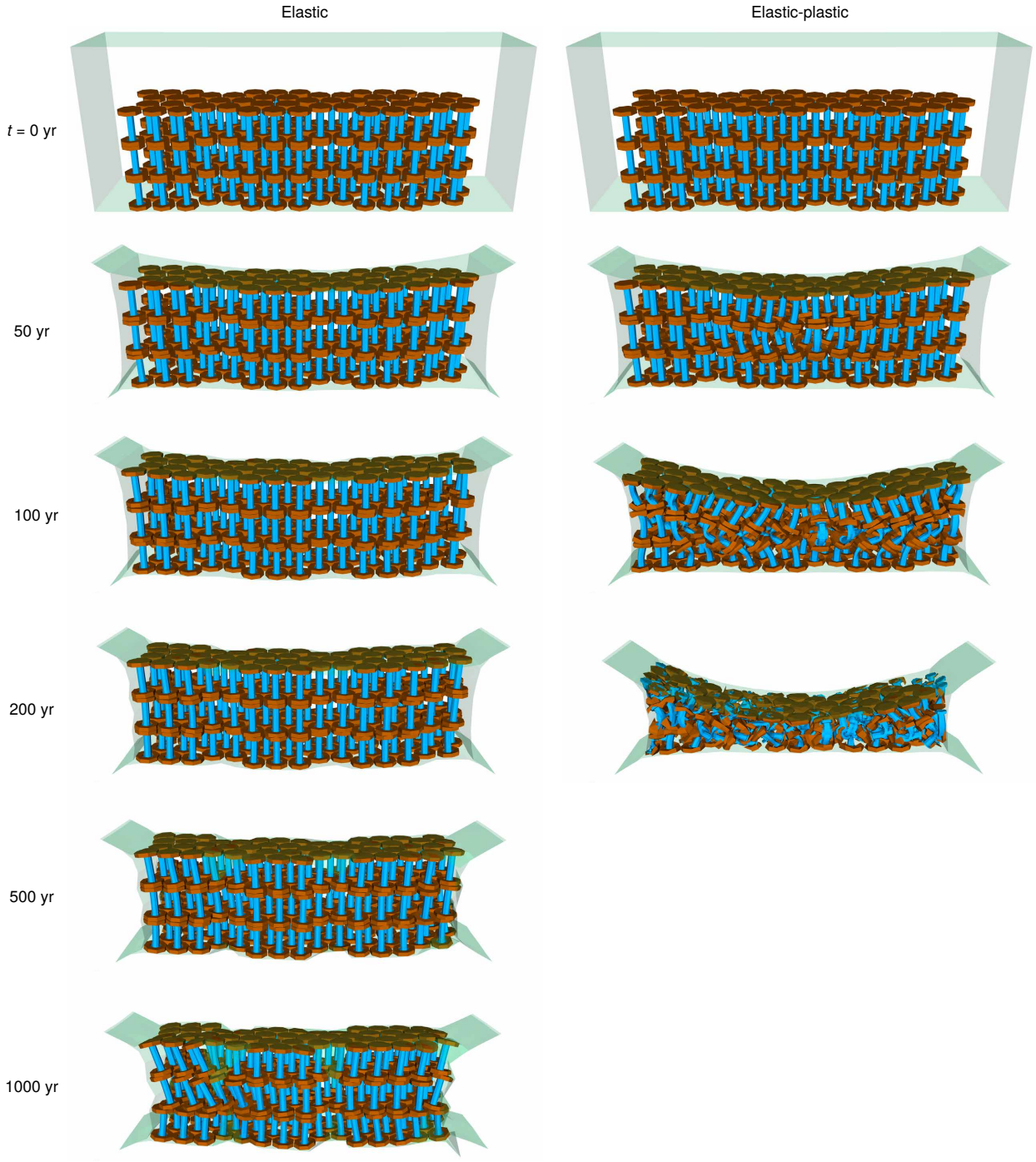


Figure 3-1. Predictions of elastic and elastic-plastic container deformation at selected instances in time. (Geology and 55-gallon drums removed for visualization purposes.)

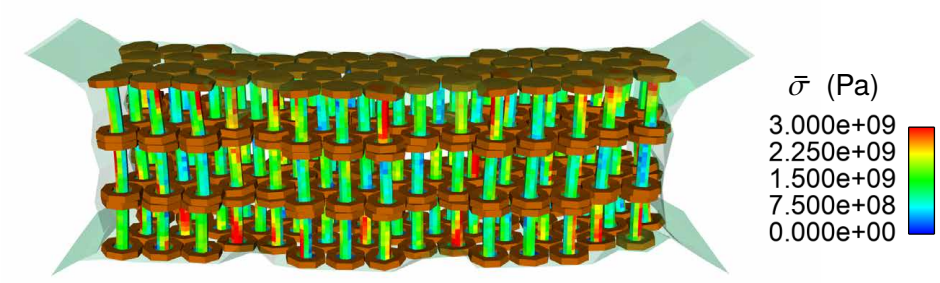


Figure 3-2. Von Mises stress distribution in stainless steel pipes for elastic containers at $t = 500$ yr.

practically certain to accommodate the vertical room closure before reaching those high stresses.

3.1.2. Elastic-Plastic

As expected, pervasive buckling occurs once the containers are allowed to plastically deform (see right side of Fig. 3-1). Column buckling begins soon after the roof comes into contact with the container stacks at the center of the room. By $t = 100$ yr, the buckling has spread to nearly every container stack in the room, and, by $t = 200$ yr, the containers are a jumbled array. Shortly before $t = 220$ yr, the simulation slowed down so much that it was impractical to continue. Elements had became so severely distorted that the critical time step size $\Delta t_{\text{cr}}^{\text{mb}}$ was prohibitively small.

Although the simulation ended prematurely, the deformed geometries for $0 \leq t \leq 200$ yr still show that the homogenization schemes discussed in Chapter 1 would have introduced some amount of error. First, the containers on the right side of Fig. 3-1 deformed in a highly heterogenous manner with signficant sliding past one another. This suggests that homogenizing one container and smearing the behavior out to all other canisters would have been highly questionable. Second, the buckling wavelength of the container stack at the front, center, of the room at $t = 50$ yr, spans the current height of the room. This indicates that homogenizing a representative volume of multiple containers would also incur some error, because scale separation between container deformation and room deformation does not exist. The error associated with the second homogenization scheme may be acceptably small, but investigations would need to demonstrate this.

3.1.3. Elastic-Plastic-Fracture

Adding fracture to the container material models allowed elastic-plastic-fracture simulation to continue on much further, because elements simply failed prior to becoming heavily deformed. Roughly half the plywood is broken at $t = 50$ yr, and all of it has disappeared by 500 yr on the left side of Fig. 3-3. Examination found that the plywood elements were meeting their failure criterion; they were not inverting. A $\sigma_{\text{f}} = 5$ MPa failure stress is quite low, so this seems plausible. On the other hand, one might expect some of the plywood to survive in the interstices between stainless steel pipes, so further investigation might be worthwhile here. Fortunately, the plywood failures seem to have little impact on the compaction process. Elastic-plastic and

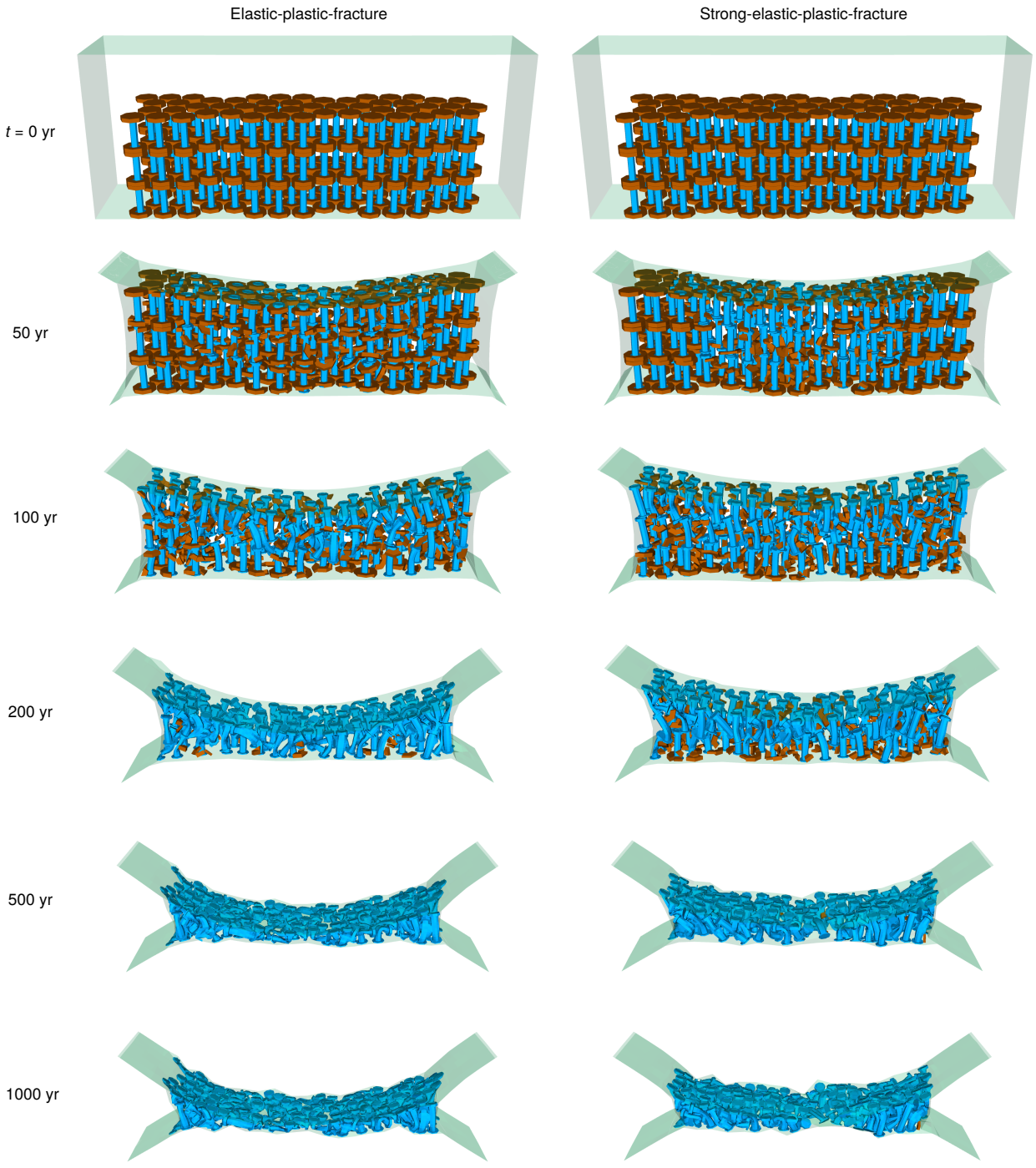


Figure 3-3. Predictions of elastic-plastic-fracture and strong-elastic-plastic-fracture container deformation at selected instances in time. (Geology and 55-gallon drums removed for visualization purposes.)

elastic-plastic-fracture compactions look fairly similar in Figs. 3-1 and 3-3, with container stacks buckling upon compressive loading from the roof and floor.

The additional 780 yr of elastic-plastic-fracture compaction show that the containers offer very little resistance to vertical room closure once the container stacks buckle. When the compacted containers finally provide enough reaction force to virtually stop the vertical closure at $t \approx 500$ yr, so much salt has flowed vertically into the room that there is no room for further horizontal closure. After $t = 500$ yr, the room shape has essentially stabilized to something resembling a tall and slender "I" turned on its side.

3.1.4. Strong-Elastic-Plastic-Fracture

The elastic containers vertically held open the room, causing non-negligible horizontal compaction rates centuries after waste emplacement, while the elastic-plastic containers collapsed under vertical pressure, shutting off the the horizontal compaction. Consequently, the strong-elastic-plastic-fracture model was conceived as an intermediate case to see whether very strong containers would collapse as well.

The right side of Fig. 3-3 shows that strengthening the container materials does not suppress the container stack buckling. In fact, the elastic-plastic-fracture and strong-elastic-plastic-fracture models deform quite similarly: they both reach approximately the same shape at 1,000 yr. Close comparison between the left and right side of Fig. 3-3 reveals that the strong containers are mangled slightly less and they hold open the vertical closure a little bit more than the nominal containers, as one might expect. In addition, a higher percentage of the strong plywood survives than the nominal plywood.

3.1.5. Quantitative Comparisons

The previous subsections provided a qualitative view of how different container models are predicted to deform, but it is worthwhile to compare them in a more quantitative fashion. Ideally, one would track the location of each block of waste, feed those locations into a nuclear criticality analysis, and compare the results based on their reactivity. That level of coordination was not achieved here. Instead, horizontal closure at room mid-height δ_h and vertical closure at room mid-width δ_v*, were used to compare different container compaction predictions. Closure is greater at these points than off-center points, so δ_h and δ_v are, in a sense, conservative. In the future, δ_h and δ_v can also be input into simple, affine, container deformation models for nuclear criticality analyses.

The closure histories for each container model are shown in Fig. 3-4. The δ_h and δ_v have been respectively normalized by L_h and L_v in Fig. 3-4 to make them into horizontal and vertical closure percentages.

*The elastic-plastic-fracture, strong-elastic-plastic-fracture, and empty room closures were averaged along the length of the room, but the elastic and elastic-plastic closures were each measured, unintentionally, only at the front of the room (one pair of points). This lack of averaging is not expected to alter the results because the closure at room mid-height and mid-width did not vary significantly along the length of the room in those simulations.

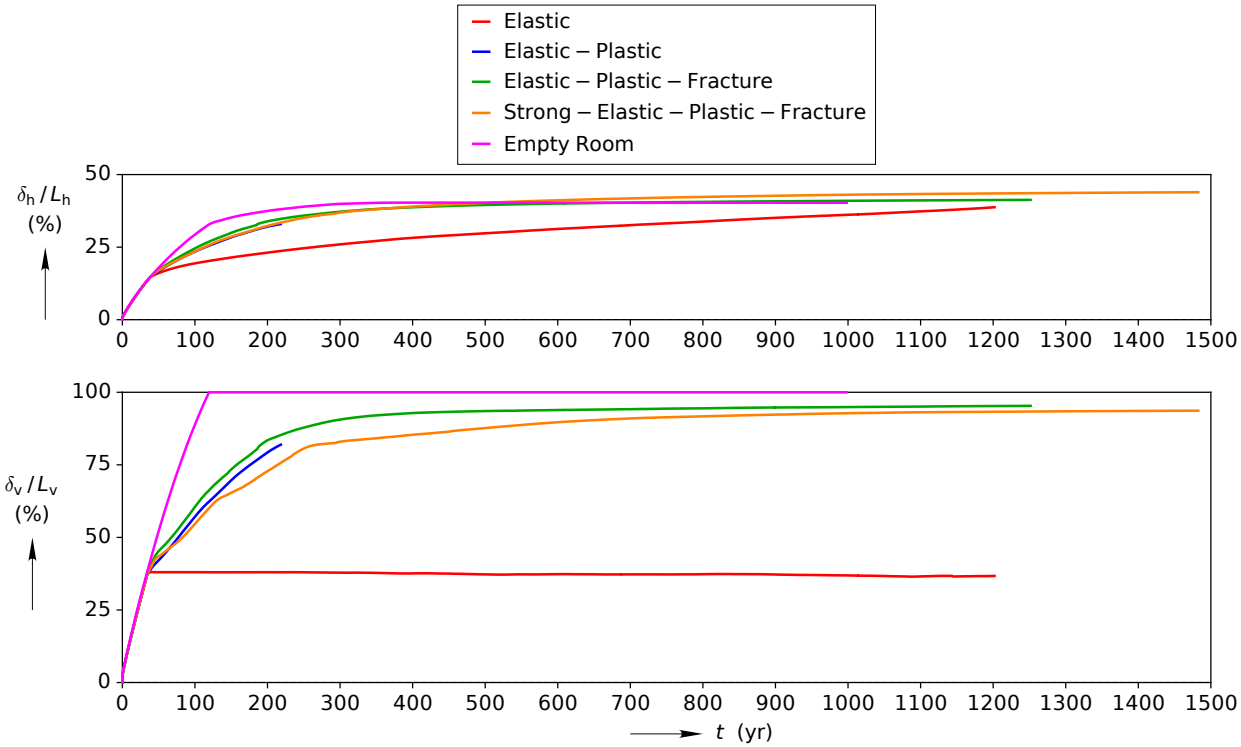


Figure 3-4. Room closure with five different container behaviors

Figure 3-4 also depicts the closure curves for an empty room. As mentioned in Section 2.2.2, an empty room represents the possibility that all canister materials degrade to zero structural stiffness. Images of an empty room closing will be displayed in Section 3.2, but it is still possible to discuss the closure curves now. Without any containers to restrain the vertical closure, δ_v/L_v quickly reaches 100 % by t = 125 yr. As one might expect, the δ_h/L_h curve exhibits a mild kink once δ_v/L_v = 100 % and exhibits another mild kink at $t \approx 310$ yr when the empty room completely closes.

The closure histories in Fig. 3-4 starkly demonstrate the impact of container stack buckling. All the closure curves agree until 35 yr, which corresponds to when the roof contacts the containers. The closure curves, especially the vertical closure, start to diverge at t = 35 yr and diverge more significantly at 46 yr. When the containers are fully elastic and excessively stiff, the vertical closure essentially halts at δ_v/L_v = 38 % upon touching the containers, while the horizontal closure continues to evolve at a monotonically decreasing rate. When the containers can plastically deform, the stacks buckle, and δ_v/L_v continues on well past 38 %. The elastic-plastic and elastic-plastic-fracture models have quite similar closure curves, which suggests that the details of the failure models chosen in Section 2.2.2 do not significantly affect the results, at least as far as t = 220 yr, when the elastic-plastic simulation ground to a halt. Interestingly, strengthening the strong-elastic-plastic-failure containers did little to delay the onset of stack buckling. This is probably because the stainless steel elastic stiffness is rather high, such that only a small amount more δ_v/L_v was sufficient to reach the stainless steel's 1.09 GPa yield strength.

Another interesting observation is the plastically deforming containers have three relatively

distinct changes in their vertical closure rates prior to asymptotically approaching the final closure values. The strong-elastic-plastic-fracture model's three closure rate changes at t = 46 yr, 132 yr, and 259 yr are the most distinctive, so they were investigated further by comparing against deformed canister images at 10 yr increments. The first vertical closure rate change is due to the initial pipe stack buckling previously discussed. This buckling allows the top layer of stainless steel pipes to be stuffed into the middle layer during t = 46 yr to 132 yr, resulting in two layers: a bottom layer and a top layer composed of twice as many stainless steel flanged pipes as the bottom layer. The second rate change at 132 yr occurs as the top layer pipe flanges slide past the bottom layer pipe flanges. The top layer descends into the bottom layer between t = 132 yr and 259 yr, to make one layer. Finally, the third rate change at t = 259 yr takes place when the deformation transitions from stuffing the top layer into the bottom layer to compressing the final layer.

The closure histories allow one to assess whether the compaction process has stabilized. The elastic container horizontal closure, for example, is far from stabilized with a rate of $\dot{\delta}_h/L_h$ = 0.015 % / yr at t = 1200 yr. The empty room, elastic-plastic-fracture, and strong-elastic-plastic-fracture simulations, on the other hand, were halted once the horizontal and vertical closure appeared to have sufficiently stabilized. Stronger containers took longer to stabilize, but even the strong-elastic-plastic-fracture case reached a horizontal closure rate of $\dot{\delta}_h/L_h$ = 0.0011 % / yr by t = 1484 yr.

Stronger containers appear to cause more final horizontal closure and less final vertical closure, but the differences are not large. The final closures and the time to reach these closures are shown in Table 3-1 for the empty room, elastic-plastic-fracture containers, and strong-elastic-plastic-fracture containers. As a reminder, gas pressure due to degradation was not included in these simulations (see Section 2.1.2). The elastic-plastic-fracture and empty room final closures only differ by δ_h/L_h = 1.0 % and δ_v/L_v = 4.7 %. The strong-elastic-plastic-fracture final closures differ from the empty room final closures by greater amounts (δ_h/L_h = 3.6 % and δ_v/L_v = 6.3 %), but the container materials are unlikely to be as strong as assumed in the strong-elastic-plastic-fracture model. The similarities show that the final closures of an empty room could be used as a computationally cheaper surrogate for a room full of CCO containers, if one can tolerate a small loss of accuracy. In addition, one can surmise that adding degradation to the CCO container materials may not substantially change the compaction predictions.

Table 3-1. Final closure predictions

Case	t (yr)	δ_h/L_h (%)	δ_v/L_v (%)
Empty Room	1,000	40.3	100
Elastic-Plastic-Fracture	1,253	41.3	95.3
Strong-Elastic-Plastic-Fracture	1,484	43.9	93.7

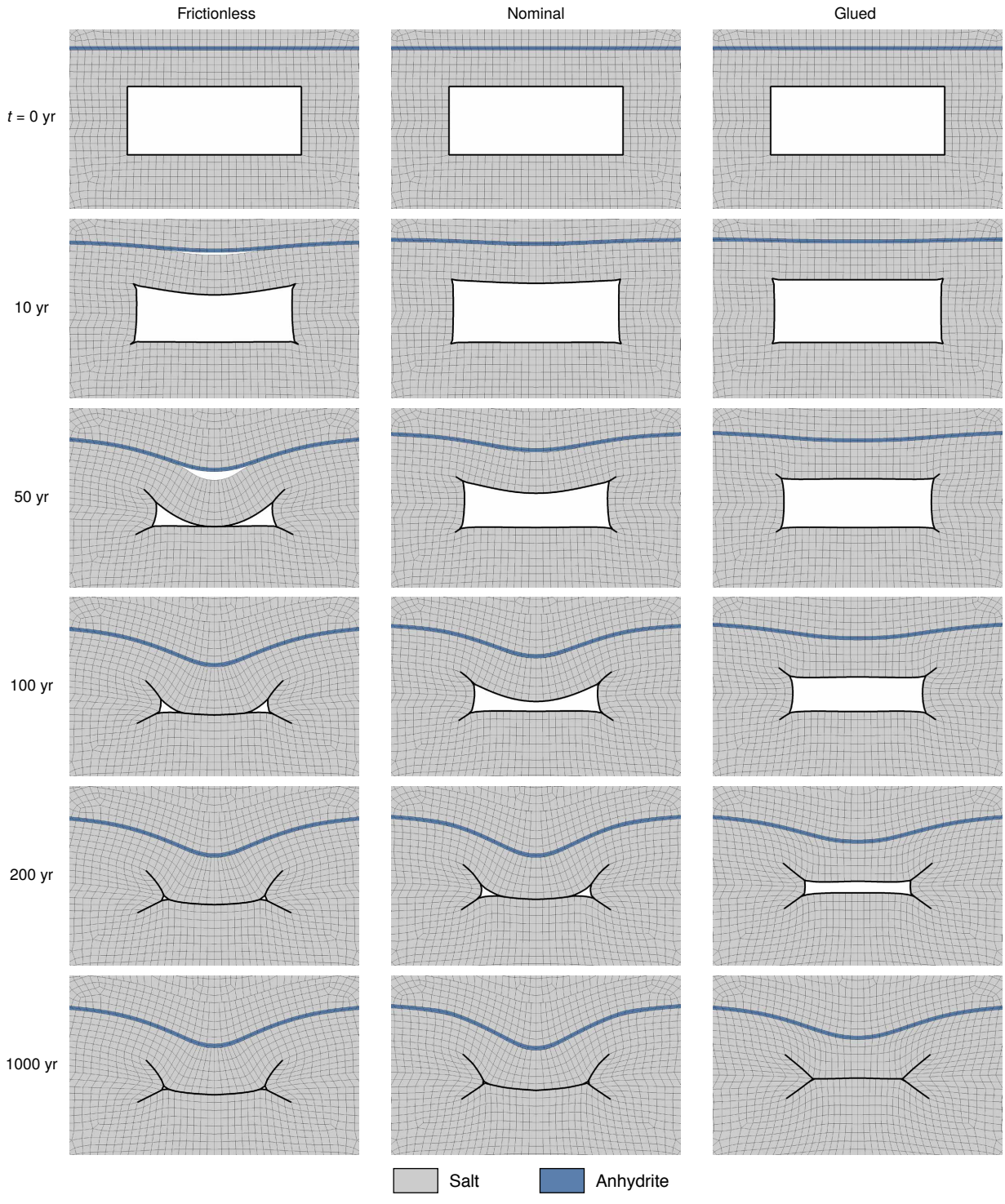


Figure 3-5. Empty disposal room shapes at various times for the three clay seam behaviors studied.

3.2. Sensitivity to Clay Seam Behavior

Thus far we have probed the sensitivity to the canister behavior, but the geomechanical model has uncertainties as well. Only the clay seam behavior was varied herein because its mechanical behavior has not been characterized, unlike the salt, anhydrite, and polyhalite. Simulations were performed using the frictionless, nominal, and glued clay seam models defined in Section 2.1.4 and Table 2-3. Given that simulations of an empty room and a room full of elastic-plastic-fracturing containers were not substantially different, the clay seam sensitivity study was carried out on an empty room for computational efficiency.

Fig. 3-5 shows that the clay seam behavior has a considerable impact on room closure. Each column in Fig. 3-5 contains a different clay seam behavior, while each row depicts a different snapshot in time. The finite elements are colored according to their rock type. Frictionless clay seams allow the roof to bow into the room early in the simulation, splitting the open space into two triangles at $t = 50$ yr. One can also see a clear separation between the salt and the thin anhydrite "a" layer above the room at $t = 50$ yr. Nominal clay seams hold back the salt beam above the room enough to delay 100 % vertical closure until after $t = 100$ yr. In addition, the separation at anhydrite "a" with the nominal clay seams is scarcely visible at $t = 50$ yr. When the clay seams are glued, the room shape maintains a roughly rectangular shape until 100 % vertical closure.

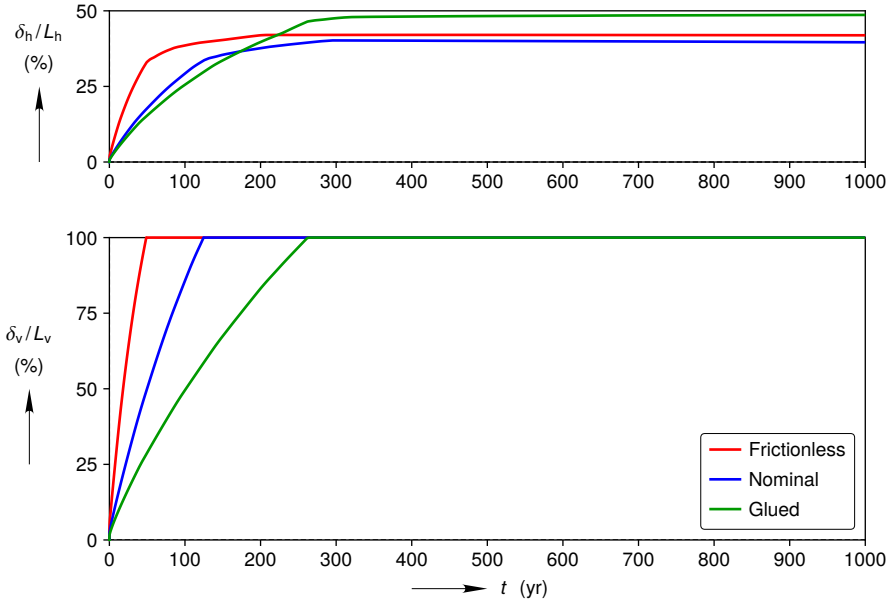


Figure 3-6. Empty room closure with three different clay seam behaviors.

Figure 3-6 shows the horizontal and vertical closure curves for empty rooms (without any gas pressure). Simulations with frictionless, nominal, and glued clay seams respectively reach 100 % vertical closure at $t = 49$ yr, $t = 125$ yr, and $t = 262$ yr, and respectively produce final horizontal closures of 41.9 %, 39.6 %, and 48.6 % at $t = 1{,}000$ yr. The vertical closure trend is as one might expect: more shear resistance and more resistance to separation results in slower vertical room closures. The horizontal closure trend, however, is counterintuitive: the glued clay seams produce

the greatest horizontal closure. As shown in Fig. 3-5, glued clay seams drag the walls of the room inward as the room closes, while the frictionless clay seams permit the roof to slump into the room and restrain further horizontal closure.

4. SUMMARY AND POTENTIAL FUTURE WORK

Once containers are placed underground in a WIPP disposal room, the surrounding rock salt compresses the containers over several centuries. A new container, called a CCO container, may have as much as 90 % more FGE of plutonium per container than previous container designs, prompting careful assessments of nuclear criticality. To support these nuclear criticality assessments, a series of geomechanical simulations was performed to predict the final compacted state of the CCO containers.

Although the collection of containers within a room has been traditionally modeled as a homogeneous solid material, the new CCO design consists of a thin-walled, 17 cm diameter, stainless steel pipe and plywood within a carbon steel 55-gallon drum. These CCO containers are structurally anisotropic and susceptible to buckling during compaction, which prompted discrete models of the containers, including the pipe and plywood within each drum. Sierra/SolidMechanics (2018) explicit dynamics capability is well suited to modeling the pervasive contact, severe plastic deformation, and material failure as the containers compress, but the timestep is typically about a microsecond. To avoid this limitation, the rock salt viscoplasticity was sped up by several orders of magnitude (while verifying that inertial effects were negligible), thereby squeezing hundreds of years into seconds.

Five different cases were considered for the container materials: elastic, elastic-plastic, elastic-plastic-failure, strengthened elastic-plastic-failure, and no containers at all (an empty room). The actual container behavior is likely somewhere between the elastic-plastic-fracture case and the empty room, because the metals will likely corrode and the wood will likely rot if water is able to reach the containers. Images of the compacted containers and the horizontal and vertical closure at the mid-height and mid-width of the disposal room, respectively, were reported. The results for each case are summarized as follows:

- The elastic containers remained largely undeformed by the room closure. They propped open the vertical closure and rearranged horizontally so that the horizontal closure continued to evolve at 0.015 % / yr after 1200 yr of closure. Further investigation found that the von Mises stresses in the elastic stainless steel pipes were unreasonably high: many locations were well above 2.5 GPa.

- The elastic-plastic containers, on the other hand, deformed in a highly heterogeneous manner because the container stacks buckled. Furthermore, the containers interacted with one another as they buckled, suggesting that the traditional homogenization scheme used on standard waste drums would have incurred significant error if it were applied to these CCO containers. Unfortunately, the elastic-plastic container simulation only reached $t = 220$ yr before the explicit time step became so small that the simulation stopped making progress.

- The elastic-plastic-fracture simulation, nevertheless, reached $t = 1,253$ yr, because the elements that reduced the time step in the elastic-plastic simulation failed and were removed. At $t = 1,253$ yr the horizontal and vertical closure had stabilized to $\delta_h/L_h = 41.3$ % and $\delta_v/L_v = 95.3$ %, respectively. Note that the elastic-plastic and elastic-plastic-fracture closure predictions were quite similar for $0 \leq t \leq 220$ yr, indicating that the fracture model did not drastically affect the results, even if it allowed the elastic-plastic-fracture case to progress further.
- The carbon steel, stainless steel, and plywood strengths in the strong-elastic-plastic-fracture case were increased beyond the values typically observed in the literature. This produced stabilized closures close to the elastic-plastic-fracture case, demonstrating that the elastic case could be disregarded and that closure is not overly sensitive to realistic strength values.
- The empty room's horizontal and vertical closure stabilized to $\delta_h/L_h = 39.6$ % and $\delta_v/L_v = 100$ %, respectively, at $t = 1,000$ yr. These values are fairly similar to the final closures observed in the elastic-plastic-fracture case, indicating that the CCO containers do not provide a large degree of structural resistance.

The behavior of the clay seams between salt strata is one of the most significant, yet uncharacterized, geomechanical factors that impacts the ratio of vertical to horizontal closure. Three clay seam cases were considered in the empty room simulations: frictionless, nominal, and glued. The vertical closure quickly reached 100 % in the first two cases, pinching off the horizontal closure to less than 42 %. The glued seams, however, delayed complete vertical closure for longer such that the horizontal closure reached $\delta_h/L_h = 49$ % before stabilizing. A value of $\delta_h/L_h = 49$ % was the greatest seen in all simulations herein.

This report has presented a simulation methodology and several quantitative predictions of how CCO containers will compact if disposed of at the WIPP. Nevertheless, it is important to emphasize that a number of uncertainties have not been quantified and this effort is attempting to predict centuries into the future with a partially validated model. A number of potential improvements were discussed in previous chapters, and they are summarized below.

1. Geomechanical Model
    a) Attempt to include rock falls.
    b) Simulate compaction on disposal room lower horizon (panels 1, 2, 7, and 8).
    c) Possibly free up computational resources by enforcing left-right symmetry with a mirror boundary condition.
    d) Vary uncertain parameters in the geologic material models.
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 model to compare against.

e) Add time-dependent behavior to the container material models.

f) Allow container materials to corrode and rot.

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.

k) Track each piece of waste to compute a spatially resolved plutonium density.

3. Numerical

a) Investigate why the plywood disappears.

b) Troubleshoot why half-containers penetrate the disposal room walls.

c) Confirm that density scaling does not impact the results.

d) Repeat the mesh, number of contact iterations, and viscoplastic rate scale factor sensitivity studies on the full container compaction model.

Keeping these caveats in mind, the simulations in this report suggest that vertical closures between $\delta_v/L_v = 93.7$ % and 100 % are possible if the containers degrade, yet horizontal closures greater than $\delta_h/L_h = 50$ % are unlikely.

BIBLIOGRAPHY

ASME (2015). *Boiler and Pressure Vessel Code, Section II-Materials-Part D-Properties-(Metric)*. American Society of Mechanical Engineers.

Biblis, E. J. (2000). "Effect of weathering on surface quality and structural properties of six species of untreated commercial plywood siding after 6 years exposure in Alabama". In: *Forest Products Journal* 50.5, p. 47.

Blandford, R., Morton, D., Snow, S., and Rahl, T. (2007). "Tensile stress-strain results for 304L and 316L stainless steel plate at temperature". In: *ASME 2007 Pressure Vessels and Piping Conference*. American Society of Mechanical Engineers, pp. 617–628.

Broome, S. T., Bronowski, D. R., Kuthakun, S. J., Pfeifle, T. W., and Herrick, C. G. (Mar. 2014). *Compaction behavior of surrogate degraded emplaced WIPP waste.* Tech. rep. SAND2014-1897. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States).

Broome, S. T., Ingraham, M. D., Flint, G., Hileman, M. B., Barrow, P. C., and Herrick, C. G. (2016). "Laboratory Testing of Surrogate Non-Degraded Waste Isolation Pilot Plant Materials". In: *50th US Rock Mechanics/Geomechanics Symposium*. American Rock Mechanics Association.

Butcher, B. M. and Mendenhall, F. (1993). *A summary of the models used for the mechanical response of disposal rooms in the Waste Isolation Pilot Plant with regard to compliance with 40 CFR 191, Subpart B*. Tech. rep. SAND92-0427. Sandia National Laboratories, Albuquerque, NM, USA.

Carrasco, R. (Sept. 2015). *Cutting head radius for Room B and Room D corners*. Personal Communication.

Chen, K., Scales, M., and Kyriakides, S. (Apr. 2018). "Material Hardening of a High Ductility Aluminum Alloy from a Bulge Test". In: *International Journal of Mechanical Sciences* 138-139, pp. 476–488.

Corona, E. and Reedlunn, B. (2013). *A review of macroscopic ductile failure criteria*. Tech. rep. SAND2013-7989. Sandia National Laboratories.

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

Deibler, L. A. (2014). *Room temperature creep in metals and alloys*. Tech. rep. SAND2014-17935. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Hoffmeyer, P. and Sørensen, J. D. (2007). "Duration of load revisited". In: *Wood science and technology* 41.8, pp. 687–711.

Holzer, S. M., Loferski, J. R., and Dillard, D. A. (2007). "A review of creep in wood: Concepts relevant to develop long-term behavior predictions for wood structures". In: *Wood and Fiber Science* 21.4, pp. 376–392.

Incorporated, P. (Mar. 2011). *Criticality Control Overpack 30-Foot Free Drop Post-Test Summary Report*. Tech. rep. 8448-R-001, Rev. 1. Ogden UT: Petersen Incorporated.

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.

Kyriakides, S. and Miller, J. (2000). "On the propagation of Luders bands in steel strips". In: *Journal of Applied Mechanics* 67.4, pp. 645–654.

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. (2015). *Criticality Control Overpack - Fabrication Drawing*. Tech. rep. CCO-DWG-001, Revision 3. Nuclear Waste Partnership LLC, Carlsbad, NM.

McNatt, J. D., Wellwood, R. W., and Bach, L. (1990). "Relationships between small-specimen and large panel bending tests on structural wood-based panels". In: *Forest Products Journal* 40.9, pp. 10–16.

Morgan, H. S. and Krieg, R. D. (June 1984). *Updated Drucker-Prager Constants for Anhydrite and Polyhalite*. Memorandum. SAND2016-3647 R.

Munson, D. E. (1997). "Constitutive model of creep in rock salt applied to underground room closure". In: *International Journal of Rock Mechanics and Mining Sciences* 34.2, pp. 233–247.

Munson, D. E. and Dawson, P. (1979). *Constitutive model for the low temperature creep of salt (with application to WIPP)*. Tech. rep. SAND79-1853. Albuquerque, NM, USA: Sandia National Laboratories.

Munson, D. E. and Dawson, P. (1982). *Transient creep model for salt during stress loading and unloading*. Tech. rep. SAND82-0962. Albuquerque, NM, USA: Sandia National Laboratories.

Munson, D. E., Fossum, A. F., and Senseny, P. E. (1989). *Advances in resolution of discrepancies between predicted and measured in situ WIPP room closures*. Tech. rep. SAND88-2948. Albuquerque, NM, USA: Sandia National Laboratories.

Munson, D. E., Jensen, A. L., Webb, S. W., and De Vries, K. L. (1996). "Brine release based on structural calculations of damage around an excavation at the Waste Isolation Pilot Plant (WIPP)". In: *Rock Mechanics Tools and Techniques: Proceedings of the 2nd North American Rock Mechanics Symposium: NARMS'96, a Regional Conference of ISRM, Montréal, Québec, Canada, 19-21 June 1996*. SAND95-1704C. Taylor & Francis US, pp. 1495–1500.

Munson, D. E., Jones, R. L., Northrop-Salazar, C. L., and Woerner, S. (1992). *Multipass mining sequence room closures: in situ data report. Waste Isolation Pilot Plant (WIPP) Thermal/Structural Interactions Program*. Tech. rep. SAND87-2687. Albuquerque, NM, USA; Livermore, CA, USA: Sandia National Laboratories.

Munson, D. E. and Morgan, H. (1986). *Methodology for performing parallel design calculations (nuclear waste repository application)*. Tech. rep. SAND85-0324. Albuquerque, NM, USA: Sandia National Laboratories.

Munson, D. E., Petney, S. V., Christian-Frear, T. L., Ball, J. R., Jones, R. L., and Northrop-Salazar, C. L. (Dec. 1992). *Geomechanical evaluation (Room G): in situ data report (December 1984 - November 1990) Waste Isolation Pilot Plant (WIPP) Thermal/Structural Interactions Program*. Tech. rep. SAND92-0582. Albuquerque, NM, USA: Sandia National Laboratories.

Munson, D. E., Torres, T. M., and Jones, R. L. (1987). "Pseudostrain representation of multipass excavations in salt". In: *The 28th US Symposium on Rock Mechanics (USRMS)*. ARMA-87-0853. American Rock Mechanics Association, pp. 853–862.

Munson, D. E. and Weatherby, J. (1993). "Two- and three-dimensional calculations of scaled in situ tests using the M-D model of salt creep". In: *The 34th US Symposium on Rock Mechanics (USRMS)*. American Rock Mechanics Association, pp. 507–510.

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).

Rechard, R. P., Day, B. A., Reedlunn, B., and James E. Bean, J. (Jan. 2022). *Improbability of Nuclear Criticality in Compacted Criticality Control Overpacks after Room Closure by Salt Creep at Waste Isolation Pilot Plant*. Tech. rep. SAND2022-0952. Sandia National Laboratories.

Reedlunn, B. (2016). *Reinvestigation into closure predictions of Room D at the Waste Isolation Pilot Plant*. Tech. rep. SAND2016-9961. Albuquerque, NM, USA: Sandia National Laboratories. DOI: 10.2172/1333709.

Reedlunn, B. (Dec. 2017). *Geomechanical Simulations of an Empty WIPP Disposal Room using Explicit Dynamics and Time-Temperature Scaling*. Memorandum. SAND2017-13593 O. Sandia National Laboratories.

Reedlunn, B. (Nov. 2018a). *Enhancements to the Munson-Dawson model for rock salt*. Tech. rep. SAND2018-12601. Albuquerque, NM, USA: Sandia National Laboratories. DOI: 10.2172/1481508.

Reedlunn, B. (2018b). "Joint Project III on the Comparison of Constitutive Models for the Mechanical Behavior of Rock Salt: Reinvestigation into Isothermal Room Closure Predictions at the Waste Isolation Pilot Plant". In: *The Mechanical Behavior of Salt IX*. Ed. by S. Fahland, J. Hammer, F. D. H. Hansen, S. Heusermann, K.-H. Lux, and W. Minkley. BGR (Federal Institute for Geosciences and Natural Resources). ISBN: 978-3-9814108-6-0.

Reedlunn, B. and Bean, J. (Oct. 2019). *Simulations of Pipe Overpack Container Compaction at the Waste Isolation Pilot Plant*. Tech. rep. SAND2019-12111. Sandia National Laboratories. DOI: 10.2172/1570309.

Reedlunn, B. and Bean, J. (May 2020a). *Further Simulations of Criticality Control Overpack Container Compaction at the Waste Isolation Pilot Plant*. Tech. rep. SAND2020-5105 CTF. Sandia National Laboratories.

Reedlunn, B. and Bean, J. (Mar. 2020b). *Impact of Properly Specifying the Clay F and Clay G Friction Coefficients in Disposal Room Closure Simulations at the Waste Isolation Pilot Plant*. Memorandum. SAND2020-3575 CTF.

Reedlunn, B. and Bean, J. (Sept. 2021). *Additional Studies of Criticality Control Overpack and Pipe Overpack Container Compaction at the Waste Isolation Pilot Plant*. Sandia Memorandum. SAND2021-11268 CTF.

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

Ross, R. J. (2010). *Wood handbook: Wood as an engineering material*. Tech. rep. FPL-GTR-190. USDA Forest Service, Forest Products Laboratory.

Rudnicki, J. W. and Rice, J. (1975). "Conditions for the localization of deformation in pressure-sensitive dilatant materials". In: *Journal of the Mechanics and Physics of Solids* 23.6, pp. 371–394.

Salzbrenner, B. and Smith, J. A. (Nov. 2018). *Uniaxial Tensile Testing of WIPP Carbon Steel Drum Material*. Personal Communication.

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).

Scherzinger, W. M. and Lester, B. T. (Mar. 2018). *Library of Advanced Materials for Engineering (LAME) 4.48*. Tech. rep. SAND2018-3231. Albuquerque, NM, USA; Livermore, CA, USA: Sandia National Laboratories.

Sierra/SolidMechanics (2018). *Sierra/SolidMechanics User's Guide*. 4.50. SAND2018-10673. 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, J. A. (Nov. 2014). *Mechanical Modeling of a WIPP Drum Under Pressure*. Memorandum. SAND2014-20083O. Sandia National Laboratories.

Sobolik, S. R. (July 2017). *Experimental Investigation of WIPP Salt Samples and Clay Seams*. Tech. rep. TP 17-03 Rev. 0. Sandia National Laboratories.

Standards, N. I. of and Technology (May 2010). *Voluntary Product Standard: PS 1-09 Structural Plywood*. Tech. rep. PS 1-09. National Institute of Standards and Technology.

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

Stone, C. M. (1997b). *SANTOS: a two-dimensional finite element program for the quasistatic, large deformation, inelastic response of solids*. Tech. rep. SAND90-0543. Albuquerque, NM, USA; Livermore, CA, USA: Sandia National Laboratories.

Wellman, G. W. (2012). *A Simple Approach to Modeling Ductile Failure*. Tech. rep. SAND2012-1343. Sandia National Laboratories.

Wilczyński, M. and Warmbier, K. (2012). "Elastic moduli of veneers in pine and beech plywood". In: *Drewno: prace naukowe, doniesienia, komunikaty* 55, pp. 47–57.

William J. Hughes Technical Center, United States Department of Defense, United States National Aeronautics and Space Administration, United States Federal Aviation Administration, and Battelle Memorial Institute Columbus Laboratories (2015). *Metallic Materials Properties Development and Standardization (MMPDS-10)*. [Washington, D.C.] : Federal Aviation Administration.

APPENDIX A. Numerical Settings

Appendices A.1 and A.2 in this chapter discuss several numerical settings utilized in the implicit quasi-static and explicit dynamic simulations, respectively. Appendix A.3 describes the finite element meshes utilized in Chapter 3.

The numerical setting sensitivity studies in Appendices A.1 and A.2 were performed on an empty room model rather than the full container compaction model. The container compaction simulations required roughly two weeks of run time, while the empty room simulations would typically require only half a day. Running the sensitivity studies on the empty room model permitted more thorough investigations in less time, but the choice, of course, assumed that the empty room model and container compaction model had similar sensitivities to each numerical setting under study.

Unless otherwise stated, all empty room simulations in this chapter utilized the following baseline settings:

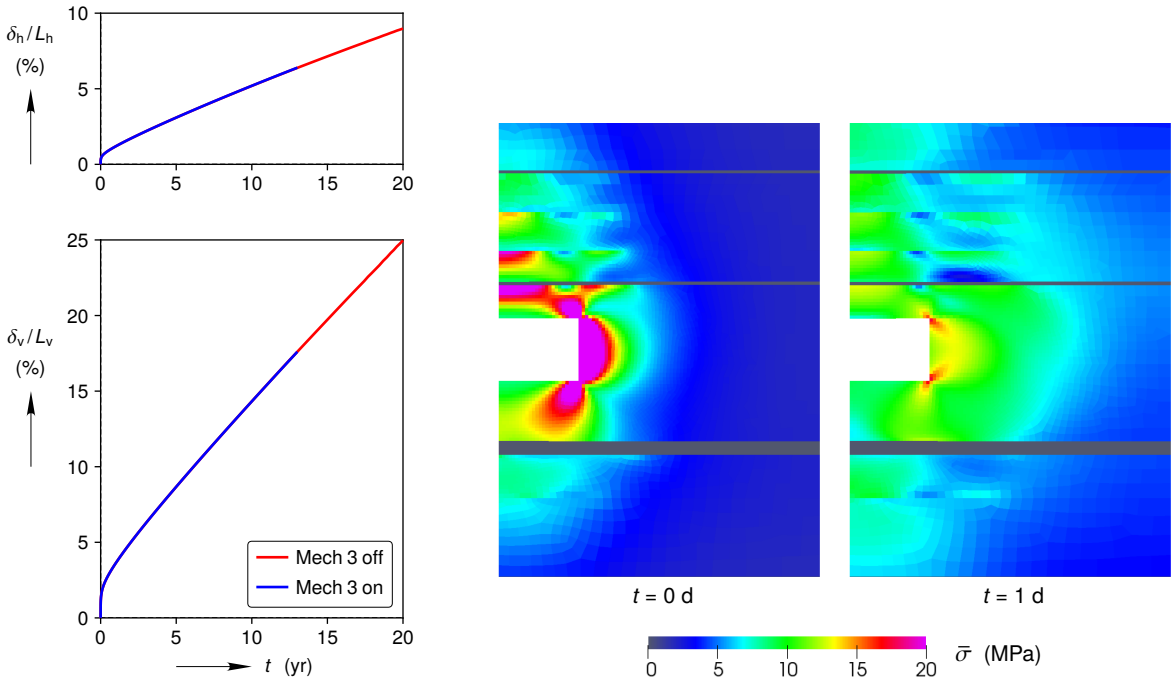
- The clay seams used the nominal parameter set in Table 2-3.
- Sierra/Solid Mechanic's selective deviatoric element, with the deviatoric parameter set to 1, substantially outperformed the default mean quadrature element in previous room closure simulations (see Fig. 8 in Reedlunn (2017)). Therefore, only the selective deviatoric element was used herein.
- The viscoplastic rate scaling used $s_1 = 2 \times 10^9$ in the explicit dynamic simulations. (The variable s_1 is defined in Appendix A.2.1.)
- Twenty contact iterations were used in the explicit dynamic simulations.
- The finite element mesh had a $L_h/48 = 0.210$ m element size that coarsened further from the room. The mesh is discussed in greater detail in Appendix A.2.3.

Examples of the Sierra/Solid Mechanics input files for the implicit quasi-static and explicit dynamic simulations can be found in Appendix D.3.

A.1. Implicit Quasi-statics

As discussed in Section 2.3.1, attempts to simulate container compaction with implicit quasi-statics were unsuccessful, so the viscoplastic rate scaling described in Section 2.3.2 was pursued. This viscoplastic rate scaling is unusual, so implicit quasi-static simulations of empty rooms were performed to create a reference solution to compare the viscoplastic rate scaled solutions against.

The implicit quasi-static simulations utilized a multi-level solver (Sierra/SolidMechanics 2018) to solve the discretized equilibrium equations. The inner loop used a conjugate gradient solver for the equilibrium equations, with the contact interactions held fixed. The outer loop utilized an augmented Lagrange algorithm to resolve the contact interactions, with the nodal displacements outside of the contact zone held fixed. The time step increment began at $\Delta t = 0.1$ s, after the virtually instantaneous excavation, and was allowed to grow by only 2 % after each successful step to ensure the MD model's differential equations were accurately integrated.

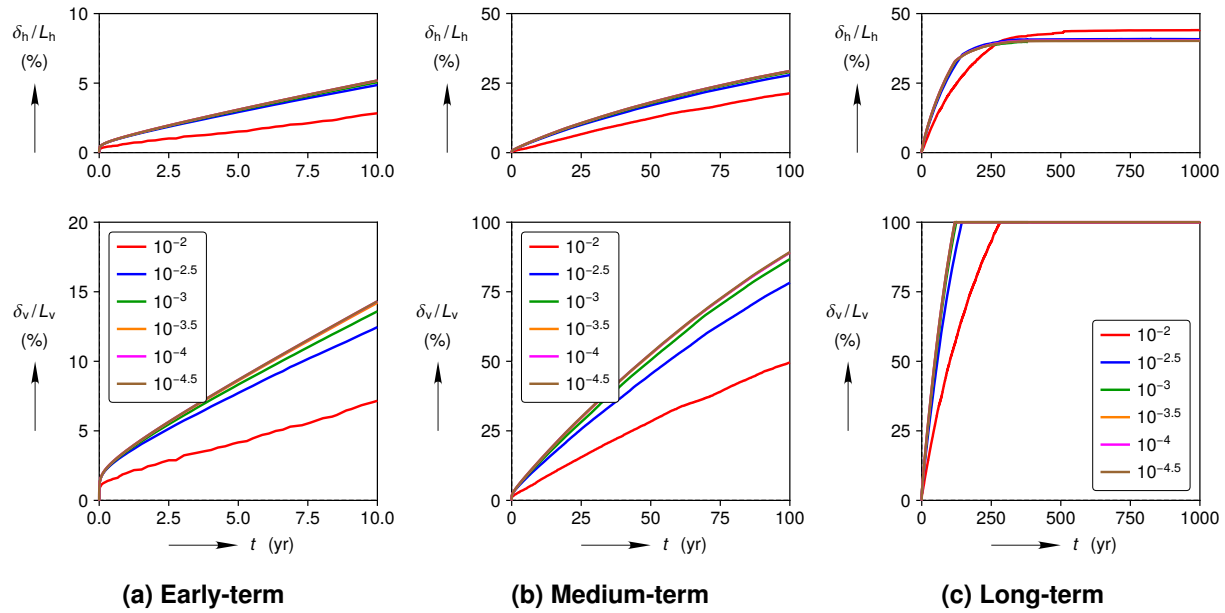


(a) Closure histories **(b) Equivalent stress fields near room**

Figure A-1. Justification for neglecting steady state mechanism 3 in further implicit quasi-static simulations.

Steady-state creep mechanism 3 in the MD model was deactivated for the implicit quasi-static simulations, because it caused simulations to fail prematurely. Mechanism 3 is only activated if the equivalent stress $\bar{\sigma}$ is above $\bar{\sigma}_g = 20.57$ MPa (Reedlunn 2018a), but it is often "accidentally" activated while the implicit solver searches for a converged solution, even if mechanism 3 does not get activated in a converged solution. If an implicit solver iteration causes a large value of $\bar{\sigma}$, say 100 MPa, then mechanism 3's hyperbolic sine term "explodes," leading to material model failures. (Explicit dynamics simulations, on the other hand, typically do not accidentally cause large $\bar{\sigma}$ values, so it was not necessary to turn off mechanism 3 for those simulations.)

To verify that mechanism 3's impact was negligible, one implicit simulation was run with steady state mechanism 3 turned on, and one implicit simulation was run without mechanism 3. As shown in Fig. A-1a, the closure predicted without mechanism 3 behaves identical to the closure predicted with all mechanisms activated, until $t = 13$ yr. The disposal room simulation with mechanism 3 failed to converge at $t = 13$ yr, while the simulation without mechanism 3 ran to completion ($t = 1000$ yr). Figure A-1b depicts the element averaged $\bar{\sigma}$ values near the room.



(a) Early-term **(b) Medium-term** **(c) Long-term**

Figure A-2. Maximum relative residual norm $\hat{R}_{\max}$ convergence study.

Mechanism 3 gets activated near the room at $t = 0$, but the $\bar{\sigma}$ field rapidly decays to less than $\bar{\sigma}_g = 20.57$ MPa in less than a day, such that mechanism 3 plays an insignificant role in the room closure.

Under the default settings, Sierra/SM accepts a solver solution as converged if

$$\frac{\sqrt{\boldsymbol{R}\cdot\boldsymbol{R}}}{\sqrt{\boldsymbol{F}^{\mathrm{re}}\cdot\boldsymbol{F}^{\mathrm{re}}}} = \hat{R} \leq \hat{R}_{\max}. \tag{A.1}$$

where $\boldsymbol{R}$ is the residual from the discretized equilibrium equations, $\boldsymbol{F}^{\mathrm{re}}$ is either the reaction force vector or the external force vector (whichever has the larger value of $\sqrt{\boldsymbol{F}^{\mathrm{re}}\cdot\boldsymbol{F}^{\mathrm{re}}}$), and $\hat{R}_{\max}$ is the user specified maximum value of the relative residual norm $\hat{R}$. In this case, the reaction force vector is bigger than the external force vector, due to the plane strain boundary conditions, so $\boldsymbol{F}^{\mathrm{re}}$ is the reaction force vector. A convergence study was conducted to properly select $\hat{R}_{\max}$. Figure A-2 depicts the horizontal closure percentage δ_h/L_h and vertical closure percentage δ_v/L_v plotted against time for six different values of $\hat{R}_{\max}$. The closure curves for $\hat{R}_{\max} = 10^{-4.5}$ and 10^{-4} overlap nearly perfectly, while the $\hat{R}_{\max} = 10^{-3.5}$ curves deviate very slightly. The $\hat{R}_{\max} = 10^{-3}$ curves may still be acceptable in some circumstances, but the $10^{-2.5}$, and 10^{-2} curves exhibit noticeable reductions in accuracy. From these results, $\hat{R}_{\max} = 10^{-4}$ was deemed sufficiently converged to be used as the quasi-static reference solution.

A.2. Explicit Dynamics

This subsection presents empty room sensitivity studies to determine the proper degree of viscoplastic scaling (Appendix A.2.1), select the number of contact iterations (Appendix A.2.2), and investigate the element size sensitivity (Appendix A.2.3). Each study varied one parameter at

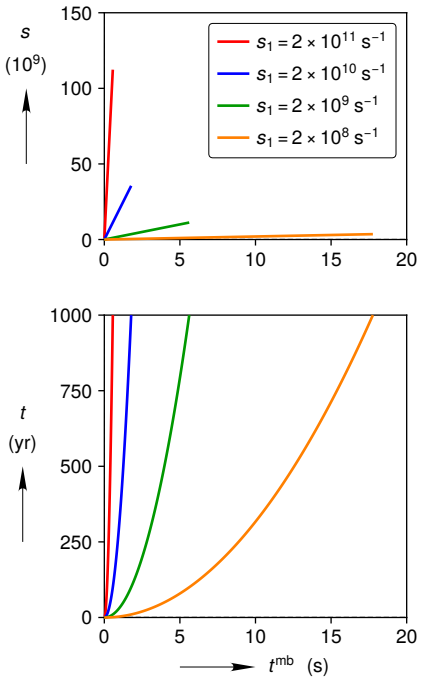


Figure A-3. Viscoplastic scaling curves.

a time, so interactions between parameters were not investigated. Appendix A.2.4 also discusses the density scaling used in the full container compaction simulations.

A.2.1. Viscoplastic Rate Scaling

Section 2.3.2 defined viscoplastic rate scaling and discussed how the scaling was implemented, but it did not describe how the s vs. t_{mb} profile was selected. Larger viscoplastic scalings result in shorter simulation run times, however they may suffer from significant inertial effects. Smaller scalings give longer run times, however they are better approximations to the quasi-static limit. One could spend a significant effort optimizing the s vs. t_{mb} scaling profile, but it would only be the optimum for a particular boundary value problem. An attempt was made to adaptively adjust s to hit a target value of kinetic energy over internal energy $E_{\text{k}}/E_{\text{i}}$ while the simulation was running, but the effort was unsuccessful. If $E_{\text{k}}/E_{\text{i}}$ would exceed the target value, s would decrease, yet $E_{\text{k}}/E_{\text{i}}$ did not decrease in time to avoid oscillations. Instead, engineering judgement was used to generate four s vs. t^{mb} scaling profiles and one was chosen based on convergence to the quasi-static reference solution discussed in Appendix A.1.

The four scaling profiles were described by

$$s = \begin{cases} s_0 & t^{\text{mb}} \le 0 \\ s_0 + s_1\, t^{\text{mb}} & t^{\text{mb}} > 0, \end{cases} \tag{A.2}$$

where $s_0 = 10^{-2}$ for all four profiles, and s_1 was varied by an order of magnitude for each profile. Figure A-3 shows the scale factor s, and the time calculated from Eq. (2.15) plotted against t^{mb}

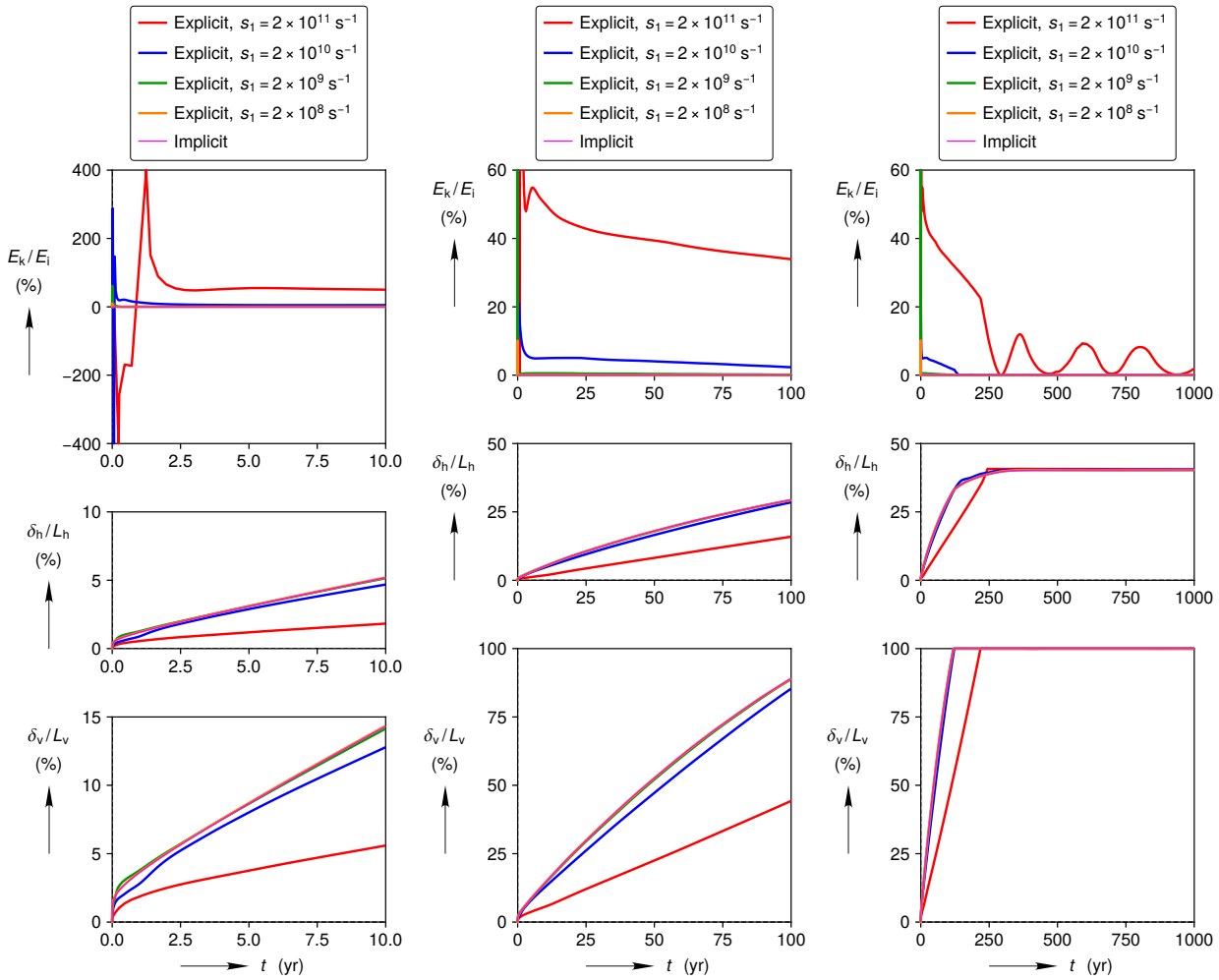


Figure A-4. Room closure and kinetic energy over internal energy histories for different viscoplastic scalings, compared against an implicit quasi-static simulation. (Note that E_k/E_i curve for $s_1 = 2 \times 10^{11}$ s^{-1} was likely under-sampled during $0 \le t \le 2$ yr.)

for the four scaling profiles. In all four cases, the scaling began with small values to avoid inertial effects during the initial transient phase of room closure, and then increased as room closure approached steady state. The four scalings covered a wide range. A value of $s_1 = 2 \times 10^{11}$ s^{-1} caused 1,000 years to be compressed into 0.562 s, while $s_1 = 2 \times 10^{8}$ s^{-1} compressed the same 1,000 years to 17.8 s.

Figure A-4 compares simulation results from the four values of s_1 against the $\hat{R}_{\max} = 10^{-4}$ quasi-static reference solution from Appendix A.1. The vertical closure, horizontal closure, global kinetic energy E_k over the global internal energy E_i are plotted against t for three different time ranges. Clearly, $s_1 = 2 \times 10^{11}$ s^{-1} is too large: the closures deviate substantially from the quasi-static results and the kinetic energy over the internal energy E_k/E_i is far from zero. Decreasing s_1 to 2×10^{10} s^{-1} still results in a E_k/E_i spike near $t = 0$, some oscillatory closure behavior in Fig. A-4a, and non-negligible differences with the reference solution in Fig. A-4b.

Reducing s_1 to 2×10^9 s^{-1} results in negligible E_k/E_i after an initial spike at $t \approx 0$ yr, and produces closure curves that almost match the quasi-static curves in Figs. A-4a and A-4b. Lowering s_1 to 2×10^8 s^{-1} decreases the initial E_k/E_i spike at $t \approx 0$ yr and leads to closure curves that agree very well with the quasi-static solution. The $s_1 = 2\times10^9$ s^{-1} curves appear to sufficiently approximate the quasi-static reference solution, so $s_1 = 2\times10^9$ s^{-1} was selected for all further explicit dynamics simulations.

A.2.2. Number of Contact Iterations

The contact enforcement routines in implicit quasi-statics and explicit dynamics have a few significant differences, so one cannot simply verify one and trust that the other will also produce accurate results. Both implicit quasi-statics and explicit dynamics used the augmented Lagrange algorithm in this work, so contact forces or momenta were balanced a-priori and contact iterations worked to resolve kinematic contact constraints*. The multi-level solver in implicit quasi-statics solves the discretized equilibrium equations anew after each contact iteration. Thus, the nodal displacements are updated throughout the finite element mesh in response to any changes in the contact interactions. The equilibrium equations (inner loop) and contact iterations (outer loop) are repeatedly solved until Eq. (A.1) is satisfied. Explicit dynamics, on the other hand, integrates the equations of motion once and then performs contact iterations. The contact iterations only update positions of nodes in the contact zone. Any force imbalances outside the contact zone that arise from the new contact interactions are turned into accelerations on the next time step. Explicit dynamics also does not have a residual to verify that the contact constraints have been accurately satisfied. It simply performs a predetermined number of contact iterations b. To summarize, implicit quasi-statics can take relatively large time steps by iterating until the equilibrium equations and contact constraints have been solved accurately enough, while explicit dynamics takes small time steps and avoids verifying solution accuracy by assuming the deformation was sufficiently small over the time step.

Given the differences in contact enforcement, a convergence study on the proper number of contact iterations b was performed to verify that explicit dynamics resolved the contact interactions accurately. The default value of b is 5, but the Sierra/Solid Mechanics manual states that 5 is often too small to accurately model frictional contact, so b was ranged from 5 to 40. Figure A-5 shows that room closures for four values of b agree well with the implicit reference solution from Appendix A.1. (All four explicit simulation closure curves deviate very slightly from the implicit reference solution at $t = 0.5$ yr, but that is due to utilizing $s_1 = 2\times10^9$ s^{-1} in the explicit simulations.)

In a similar study, Reedlunn (2017) found the closure was mildly sensitive to b, such that $b \geq 20$ was required to obtain converged closure curves for $t \leq 100$ yr and the horizontal closure did not appear converged with even $b = 80$ for $t \geq 100$ yr. He attributed the sensitivity to faceting of the

*Before the augmented Lagrange algorithm was implemented in Sierra/Solid Mechanics, the contact enforcement algorithm satisfied the kinematic constraints a-priori and iterated to balance the momentum. Thus, Sierra/Solid Mechanics refers to contact iterations as `momentum balance iterations` for historical reasons, but `momentum balance iterations` actually resolve the kinematic contact constraints when the augmented Lagrange algorithm is employed.

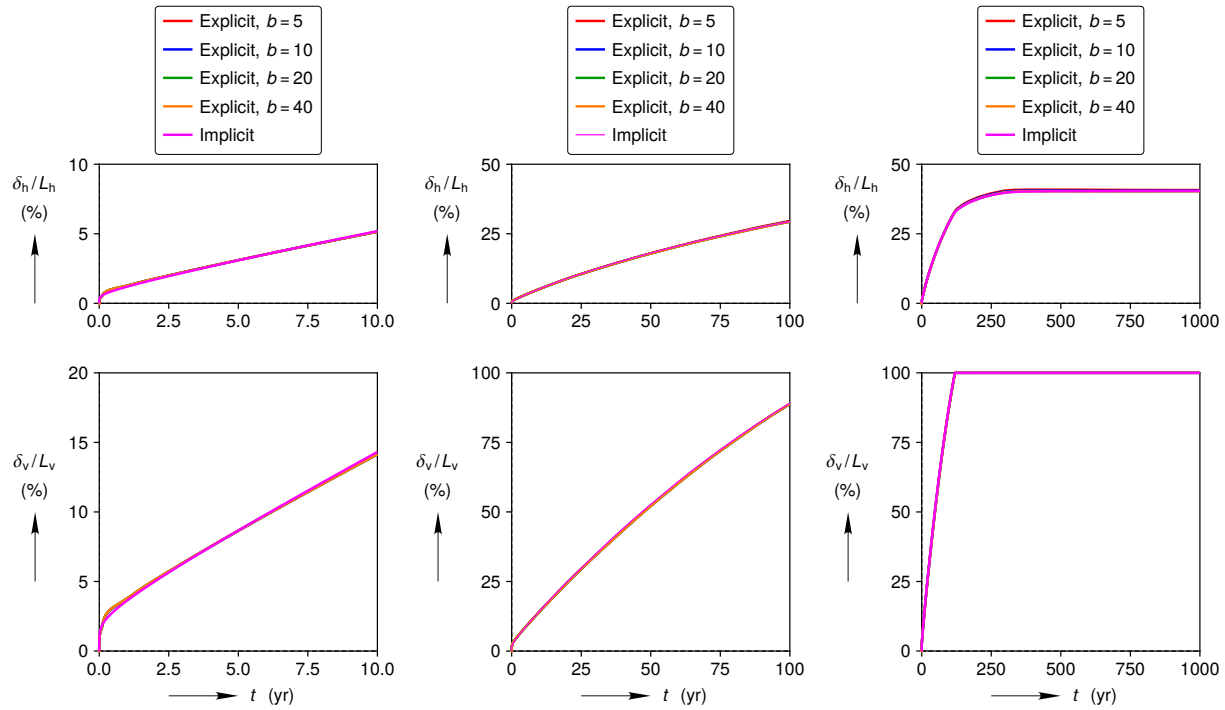


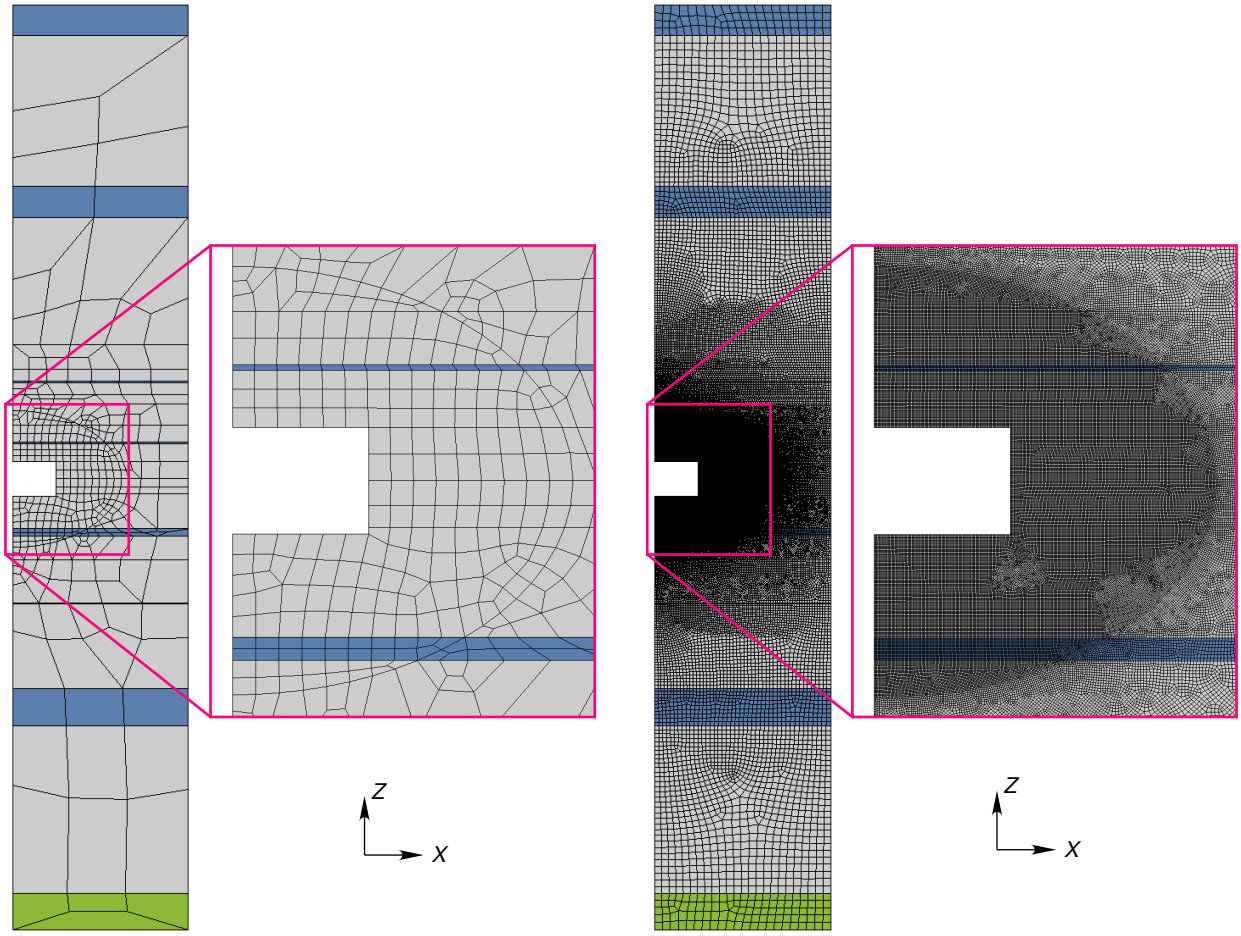
Figure A-5. Room closures histories for different numbers of explicit dynamic momentum balance iterations.

room surface by the relatively coarse $L_h/24$ element size. Herein, the element size was $L_h/48$ and the sensitivity to b is practically negligible. Examining Fig. A-5c closely, one can see the horizontal closure for $t \geq 125$ yr changes very slightly between $b = 5$ and 20, but the $b = 20$ and 40 closure curves lie on top of one another. The computational cost of using $b = 20$ rather than $b = 5$ is not severe, and b may be more important once the containers are included in the explicit dynamics simulations. In conclusion, $b = 20$ was used in all further simulations, although $b = 5$ may be sufficient.

A.2.3. Element Size

The last numerical setting to be varied was the element size. Figure A-6a exhibits the coarsest mesh with 6 elements across the half-width of the room ($L_h/12$ element size) and Figure A-6b exhibits the finest mesh with 96 elements across the half-width of the room ($L_h/192$ element size). The meshes were generated using Cubit (Cubit Team 2017). An example Cubit journal file can be found in Appendix D.2.1.

The impact of element size on empty disposal room closure is displayed in Fig. A-7. Refining the mesh from a $L_h/12$ element size to a $L_h/96$ element size causes the closure curves to evolve noticeably, but refining the mesh further to $L_h/192 = 0.052$ m element size produces negligible change from $L_h/96 = 0.105$ m element size. This result is consistent with the Room D



(a) $L_{\mathbf{h}}$ / 12 **element size** **(b)** $L_{\mathbf{h}}$ / 192 **element size**

Figure A-6. Coarsest and finest finite element meshes utilized in geomechanics mesh convergence study.

convergence study in Section 2.2 of Reedlunn (2016), where the closure curves from the 0.057 m element size practically laid on top of the closure curves from the 0.115 m element size.

Although a $L_{\mathrm{h}}/96$ element size is needed to obtain a mesh converged solution, practical considerations forced further simulations to utilize a larger element size. While element sensitivity clearly exists in the short and medium term (Figs. A-7a and A-7b), sensitivity is reduced in the long term (Fig. A-7c). Nuclear criticality concerns are likely highest once δ_{v} and δ_{h} evolve to their maximum values. At $t = 1{,}000$ yr, δ_{v} is identical for all five element sizes and δ_{h} only differs by 2.2 % between the $L_{\mathrm{h}}/24$ and $L_{\mathrm{h}}/192$ element sizes. Thus, an element size of $L_{\mathrm{h}}/24$ was chosen to balance accuracy with ease of computation.

A.2.4. Density Scaling

The numerical stability of the explicit dynamic integration scheme in Sierra/SolidMechanics (2018) requires the time step size Δt^{mb} to be beneath a critical time step size $\Delta t^{\mathrm{mb}}_{\mathrm{cr}}$. In

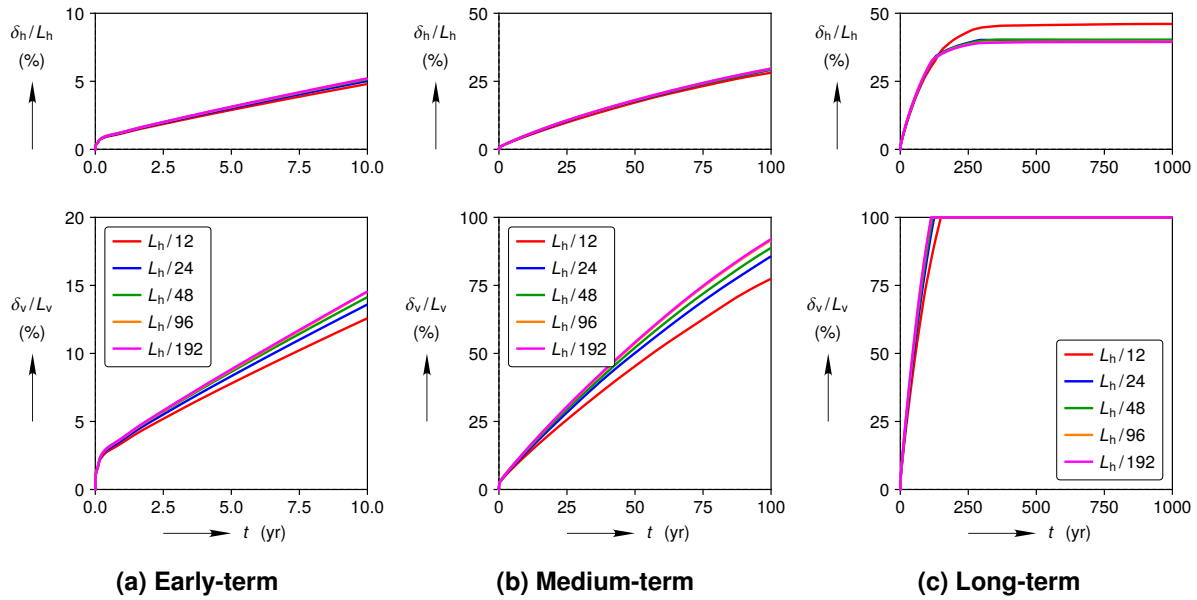


Figure A-7. Convergence of horizontal and vertical closure histories as the element size is decreased.

one-dimension, this critical time step is calculated as

$$\Delta t_{\text{cr}}^{\text{mb}} = \min\left(\frac{h}{c}\right), \tag{A.3}$$

where min(·) is the minimum over the current (deformed) finite element mesh, h is the element length, and c is the wave speed. By default, Sierra/SolidMechanics (2018) conservatively estimates the wave speed by assuming linear elastic behavior, such that

$$c = \sqrt{\frac{E}{m\rho}}, \tag{A.4}$$

where m is the density scale factor. Typically, the scaled density $\rho_{\text{s}} = m\rho$ is equal to the "physical" density ($m = 1$), but Sierra/SolidMechanics (2018) allows one to scale the density to increase $\Delta t_{\text{cr}}^{\text{mb}}$. The user specifies a desired $\Delta t_{\text{cr}}^{\text{mb}}$ and m is selected according to Eqs. (A.3) and (A.4) for each element. Note that ρ_{s} is used when balancing momentum, but ρ is used for body forces, such as forces due to gravity[†].

An element size of $L_{\text{h}}/24$ for the geomechanical model resulted in an initial value of $\Delta t_{\text{cr}}^{\text{mb}} = 4.02$ µs without any density scaling. Combining the container and geomechanical models caused $\Delta t_{\text{cr}}^{\text{mb}}$ to drop by a factor of 3.3× due to a small controlling element in the top flange of the stainless steel pipe. It was decided to scale the container densities to meet a desired $\Delta t_{\text{cr}}^{\text{mb}} = 4.0$ µs. This desired time step scaled the controlling element's density by $m = 10.9\times$ and scaled other container elements to lesser degrees. The scale factor, averaged over the volume of the top flange, was $\bar{m} = 5.3$. Note that the m values likely increased somewhat as h decreased due deformation later in the simulation. The container deformation was assumed to be sufficiently quasi-static that this density scaling did not affect the results.

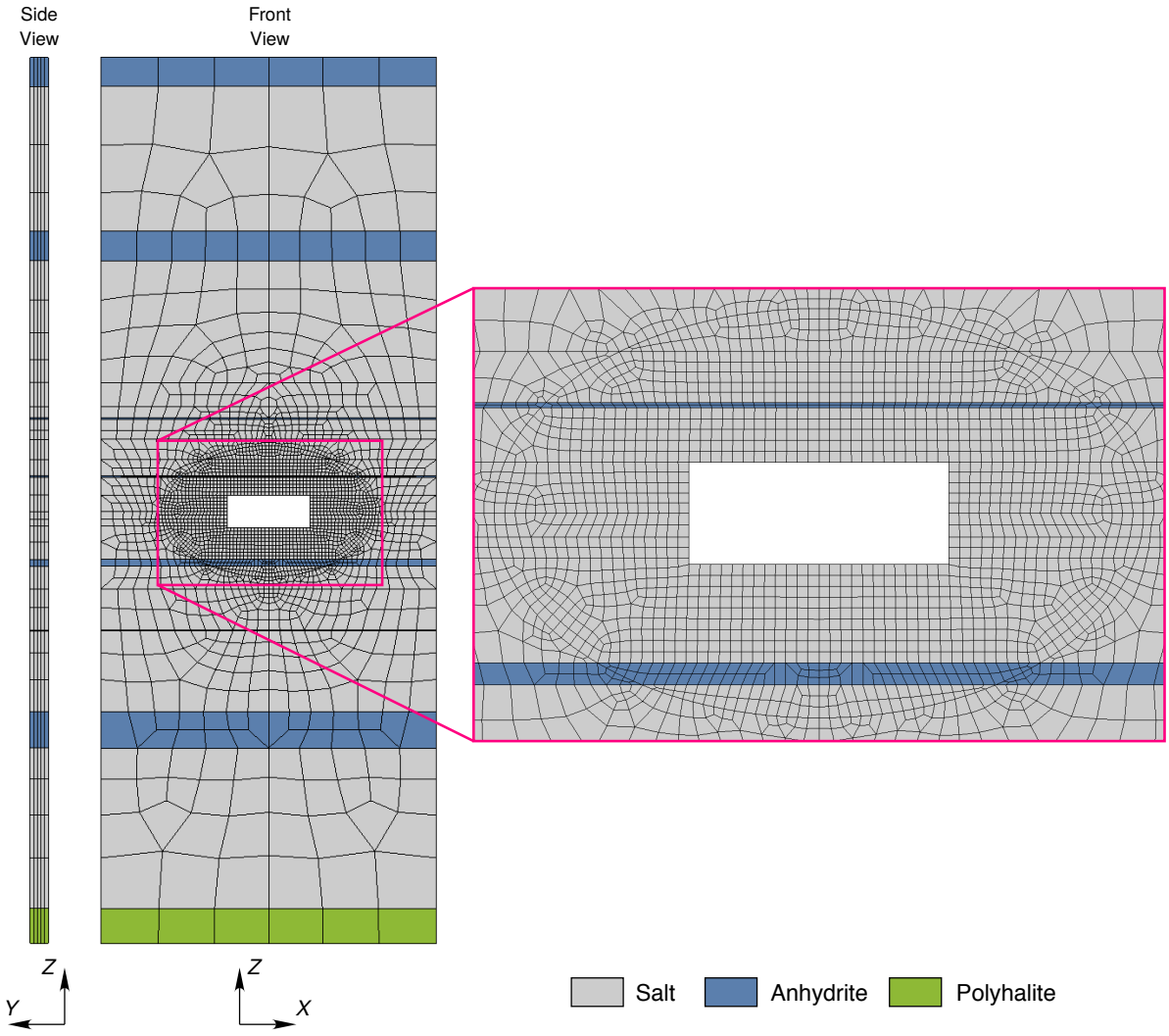
[†]A future version of the Sierra/SolidMechanics (2018) manual will make it clear that ρ, not ρ_{s}, is used to calculate body forces due gravity and centripetal acceleration.

A.3. Finite Element Meshes

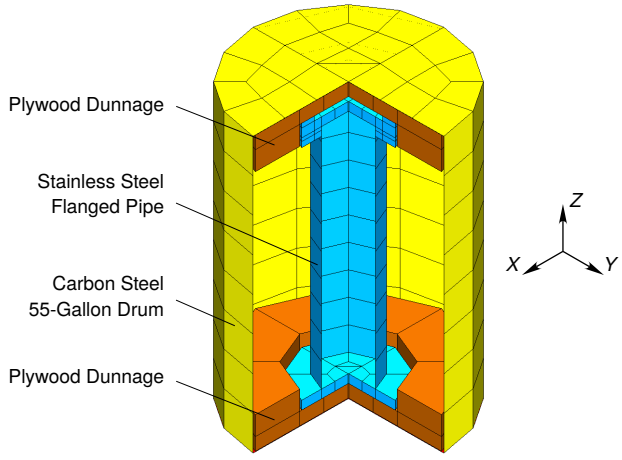
Fig. A-8a depicts the mesh of the geomechanical model utilized in the container compaction simulations in Chapter 3. Selective deviatoric elements were used throughout the geomechanical model. As mentioned in Section 2.1.2, the container compaction simulations did not take advantage of the disposal room's left-right symmetry or constrain the deformation to be plane strain, so the mesh in Fig. A-8a includes the full width D_{h} between disposal rooms and has a depth of $L_{\mathrm{d}} = 2.15$ in the Y-direction. The element width and height near the room were $L_{\mathrm{h}}/24$, while the thickness in the Y-direction was $L_{\mathrm{d}}/5$ throughout the mesh. The empty room meshes in Chapter 3 utilized the same $L_{\mathrm{h}}/24$ element width and height near the room, but only one element through the thickness to enforce the plane strain constraint. An example Cubit (Cubit Team 2017) journal file can be found in Appendix D.2.1.

Fig. A-8b shows the container finite element mesh corresponding to the idealized container geometry in Fig. 2-3. Solid, selective deviatoric, elements represented the plywood and pipe flanges, while Sierra/SolidMechanics (2018)'s default, Belytschko-Tsay, shell elements represented the 55-gallon drum and pipe. The container mesh was quite coarse to attain reasonable simulation run times. An example Cubit (Cubit Team 2017) journal file can be found in Appendix D.2.2.

Although a mesh convergence study was performed on the closure of an empty room (see Appendix A.2.3), another study was not performed on the full container compaction model. The empty room mesh convergence study should still be relevant for the geomechanical portion of the full container compaction model, presuming an empty room and a room filled with containers have similar spatial gradients inside the rock mass. Had a mesh convergence study been performed on the full container compaction model (or on a single container), finer container meshes would likely exhibit greater container compaction.



(a) Geomechanical finite element mesh



(b) Container finite element mesh (quarter wedge removed for visualization purposes)

Figure A-8. Finite element meshes utilized in container compaction simulations.

APPENDIX B. Preliminary Validations of the Geomechanical Model

Reedlunn (2018b) previously showed that the fundamental portions of the geomechanical model described in Section 2.1 can reasonably predict the vertical closure rate of Room D. Room D, however, had a 1-to-1 aspect ratio instead of the 2.5-to-1 aspect ratio of the disposal rooms, and Room D was at a higher elevation than the disposal rooms. Consequently, this chapter details further validation of the geomechanical model against Room G and Room Q's closure measurements. Predictions of Room D closure are also included in this chapter because Reedlunn (2018b) only reported the vertical closure of Room D, yet both vertical and horizontal closure are important for nuclear criticality concerns. Although the predictions in this chapter are somewhat preliminary, they do show the model can adequately predict early-term closure rates of rooms.

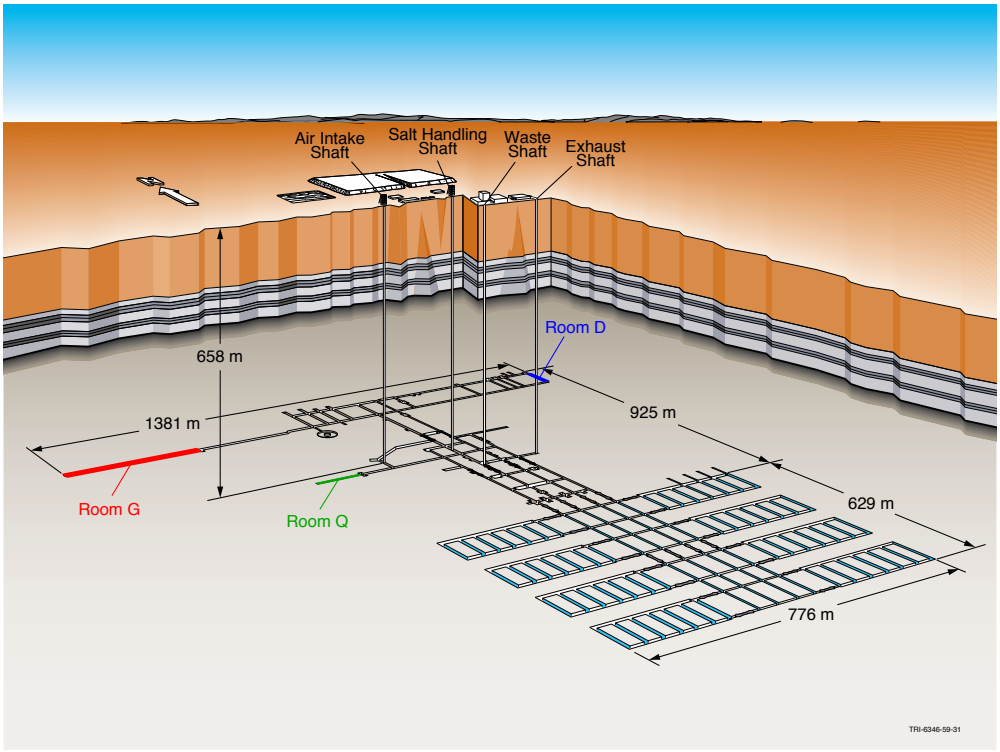


Figure B-1. Locations of Rooms D, G, and Q at the WIPP

Rooms D, G, and Q were in-situ experiments at the WIPP where the room closure was measured immediately following excavation (Munson, Jones, et al. 1992; Munson, Petney, et al. 1992). They were long isolated rooms (see Fig. B-1), making them well suited for model validation. Rooms G and Q were mined at a similar elevation to disposal panels 1, 2, 7, and 8, which was significantly below Clay G, as shown in Fig. B-2. Room G had a 2-to-1 aspect ratio, while Room Q had a circular cross-section.

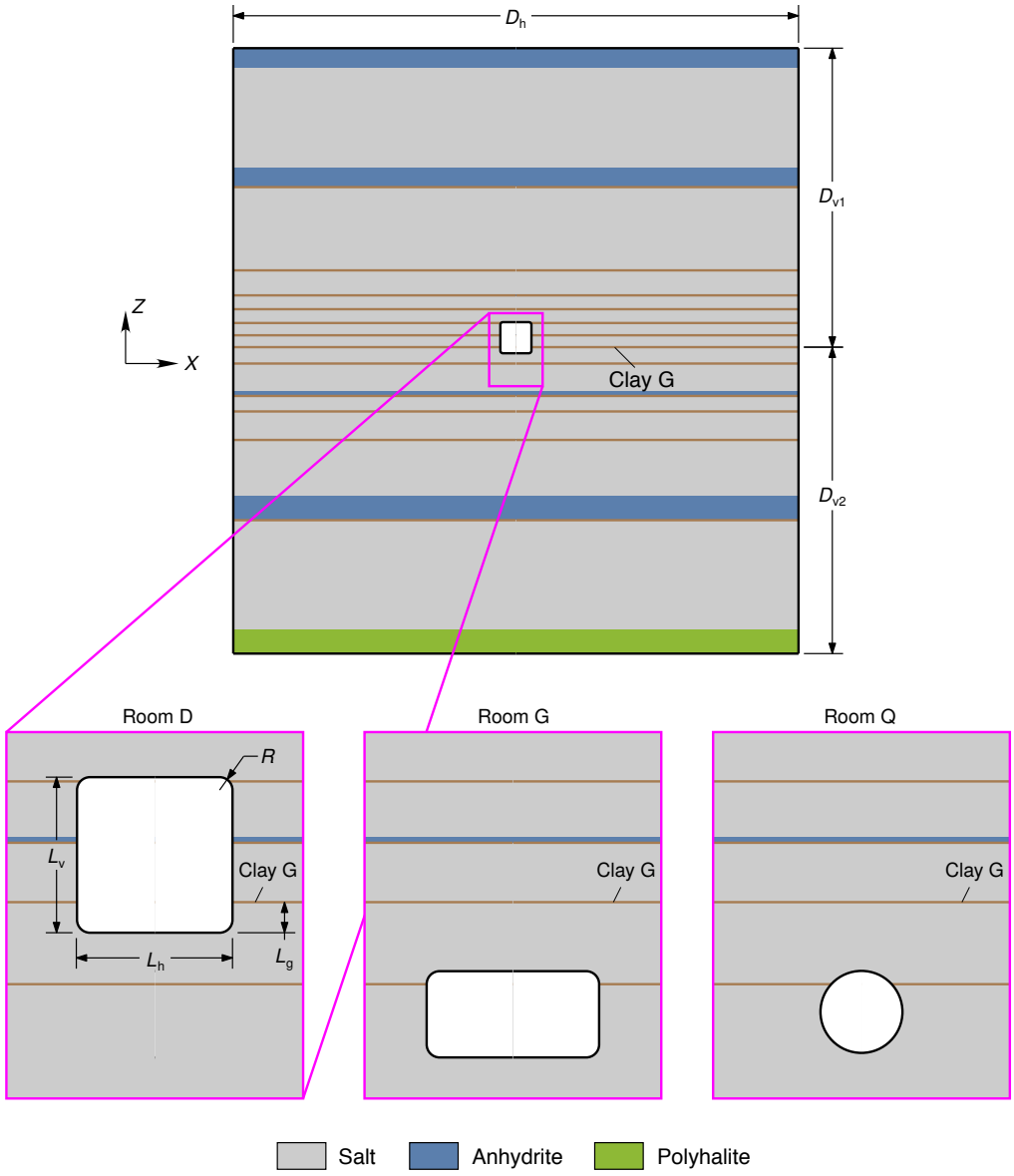


Figure B-2. Idealized plane strain geometry of Rooms D, G, and Q.

The Room D, G, and Q closure simulations utilized the essential components as the geomechanical model described in Section 2.1, but a few details were different:

- The disposal room simulations sometimes varied the clay seam behavior. Here, the clay seams adopted the nominal values of $\mu_c = 0.2$ and $\sigma_c = 0$.
- The width of the domain was increased to $D_h = 100.0$ m to better approximate the isolation of Rooms D, G, and Q, as shown in Fig. B-2.
- The room geometry and elevation were changed to the values shown in Table B-1 for each simulation.

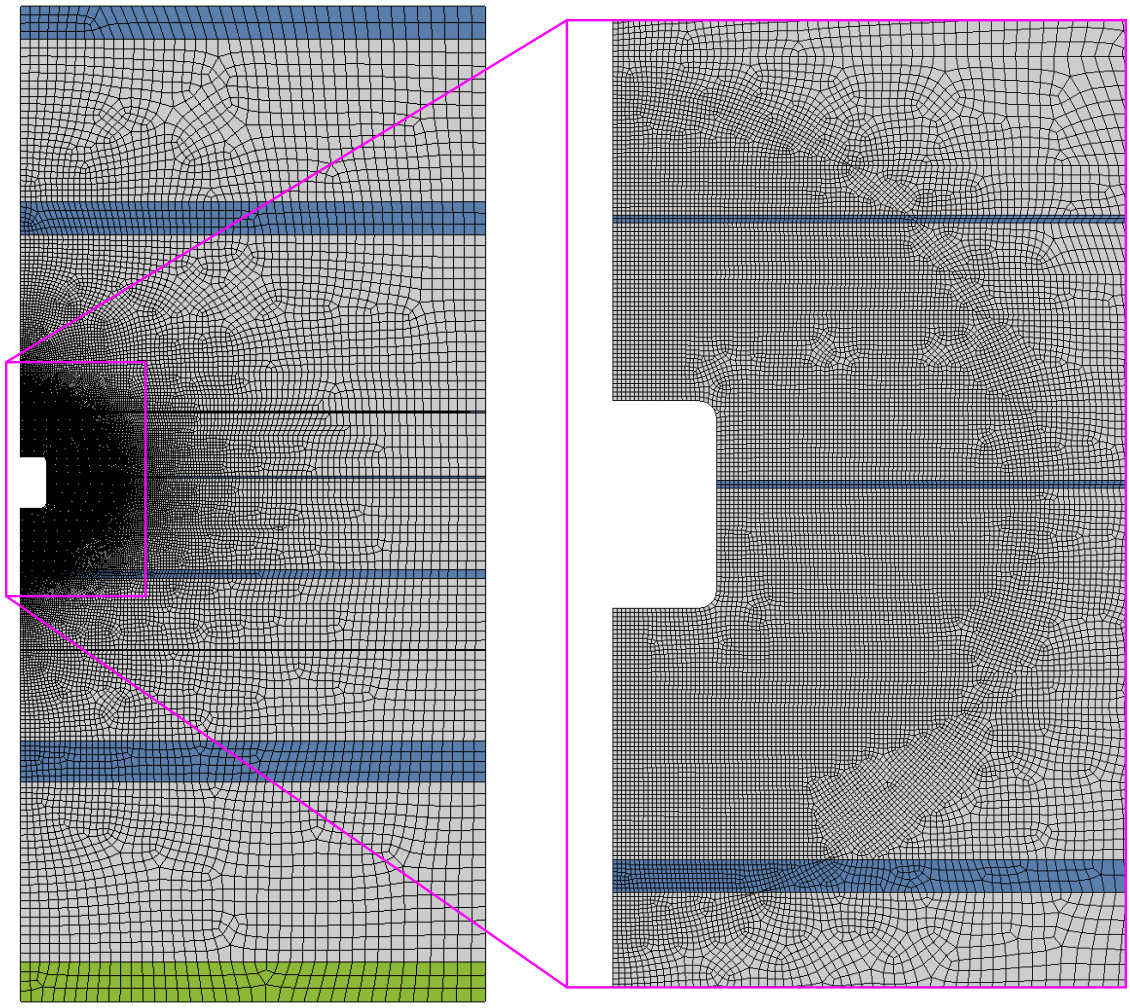


Figure B-3. Room D finite element mesh.

- The corners of Rooms D and G were rounded to a radius of $R = 0.46$ m to reflect the actual as-mined geometry (Carrasco 2015).
- The boundary conditions were the same as those in Fig. 2-2, except the left half of the domain was replaced with a symmetry boundary condition.

The numerical settings were similar to the implicit quasi-static simulations in Appendix A.1, except:

- Steady-state creep mechanism 3 was left on.
- The max relative residual was $\hat{R} = 10^{-4}$ or smaller.
- The time step was only allowed to grow by 1 % with each successful step.
- The element sizes were 0.115 m, 0.127 m, and 0.120 m near Rooms D, G, and Q, respectively.

The Room D, G, and Q meshes were similar, so only the Room D mesh is shown in Fig. B-3. See Appendix D.3.1 for an input file resembling the input files used in the Room D, G, and Q simulations.

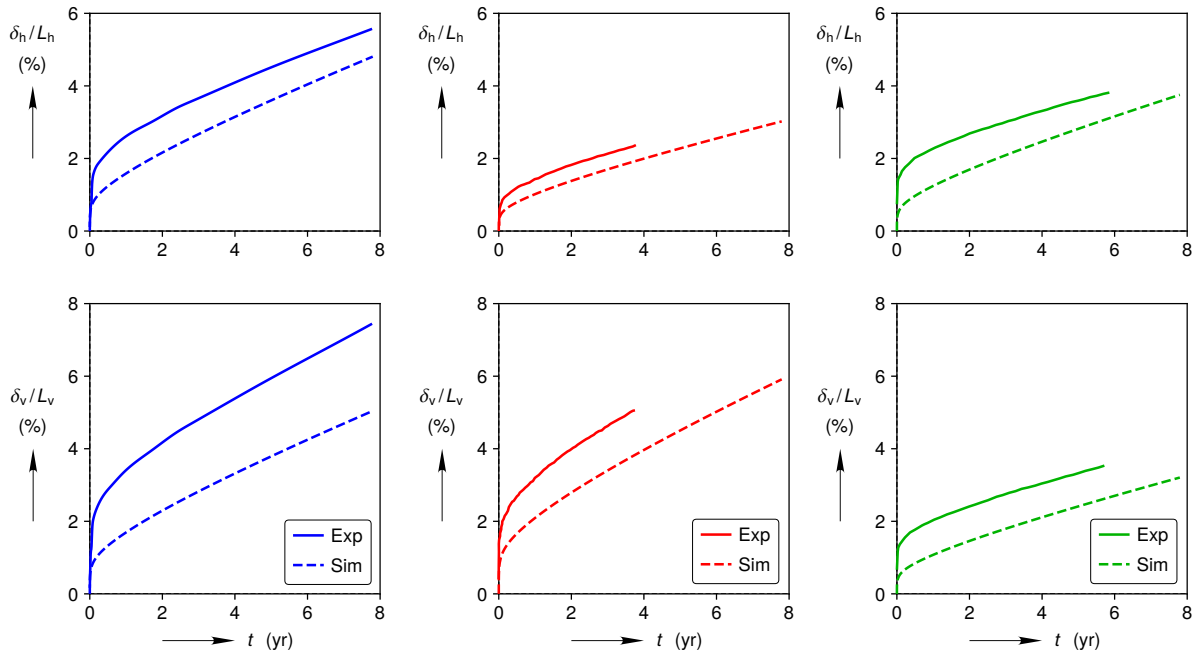
Table B-1. Rooms D, G, and Q dimensions (in meters).

Dimension	Room D	Room G	Room Q
L_h	5.50	6.10	2.90
L_v	5.50	3.05	2.90
L_g	1.08	5.49	5.33

The simulations of Rooms D, G, and Q's closure are compared against the experimental measurements in Fig. B-4. The Room D closure measurements come from a recent reconstruction of the raw, handwritten, closure measurements*. The measurements of Rooms G and Q closure were digitized from Fig. 15 in Munson (1997) and Fig. 4 in Munson et al. (1996)†. Although the horizontal and vertical closures are under-predicted, the closure *rates* of all three rooms are satisfactorily captured in Fig. B-4. Further research may yet improve the predictions of closure during the first decade, but accurate prediction of the closure rates is far more important when models are used to extrapolate forward by hundreds of years. Room G is the most similar to the disposal rooms at the WIPP, and the model, coincidently, does the best job at predicting Room G's closure rates. Any discrepancies between the predictions and measurements of Room G's closure rates are within the measurement error. Furthermore, the vertical-to-horizontal closure ratio r is of particular interest for nuclear criticality concerns and the model underpredicts Room G's r by only 7 % at 3.7 yr.

*Documentation of the reconstruction is still in progress, but it is worth noting that the first 3.7 yr of reconstructed values are consistent with the Room D closure curves in Munson et al. (1989).

†Fig. 16 in Munson (1997) also contains Room Q closure curves, but the closure data for 28.0 m east of centre and 34.4 m west of centre are significantly different. The closure measurements at two stations along the length of the room should be very similar for the room to be treated as plane strain. Further investigation found that the same data was plotted in Fig. 2 in Munson and Weatherby (1993), judging by the locations of the individual data points, and the two stations have nearly identical closure curves in that figure. Fig. 4 in Munson et al. (1996) also depicts the closure of Room Q at a station 3.6 m west of centre and the curves agree well with Fig. 2 in Munson and Weatherby (1993). Thus, it was concluded that the closure curves for the 34.4 m west of centre station in Fig. 16 of Munson (1997) were scaled incorrectly.

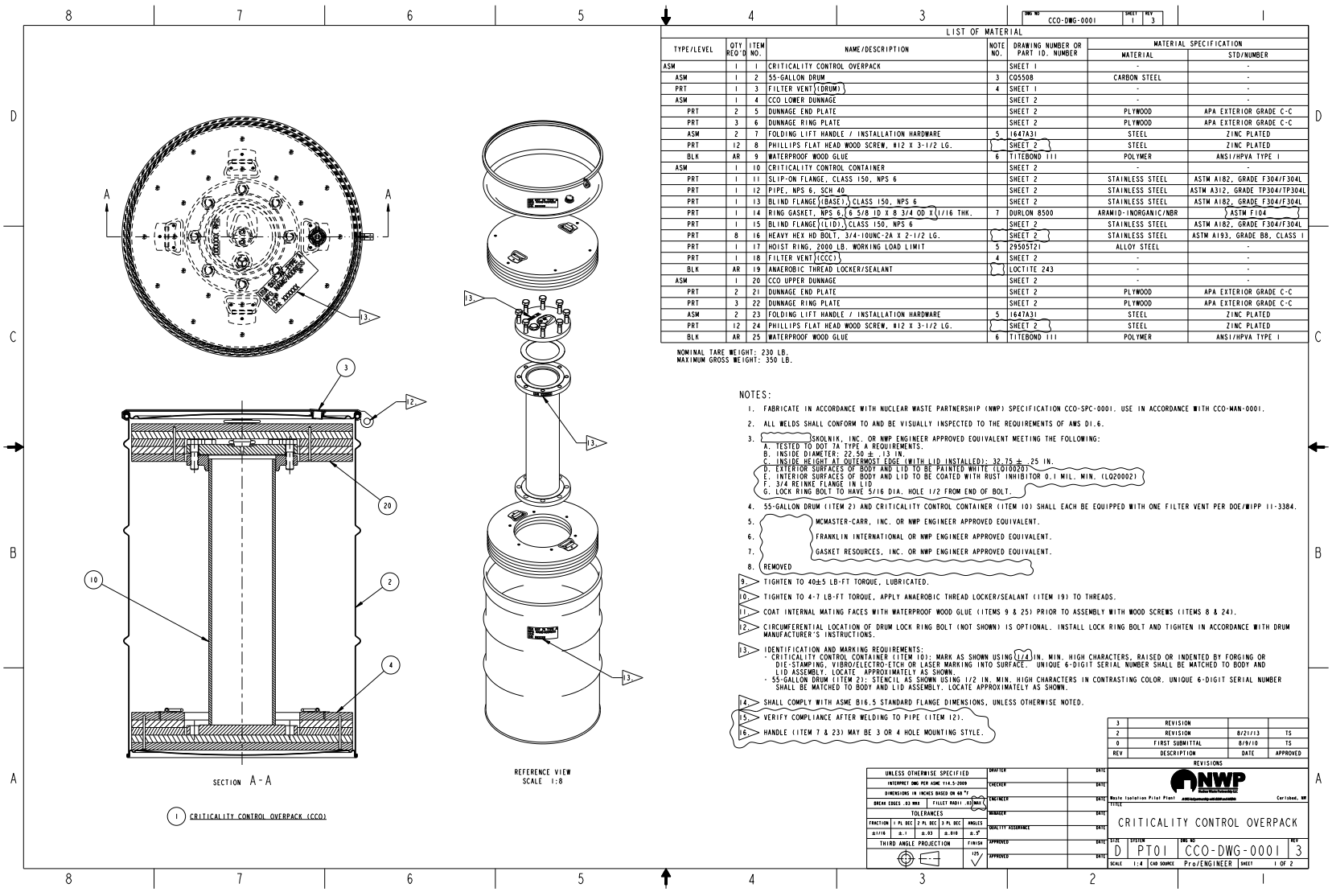


(a) Room D **(b) Room G** **(c) Room Q**

Figure B-4. Preliminary validations of the geomechanical model

APPENDIX C. Container Data

C.1. Technical Drawings



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	CRITICALITY CONTROL OVERPACK		SHEET 1	-	-
ASM	1	2	55-GALLON DRUM	3	CQ5508	CARBON STEEL	-
PRT	1	3	FILTER VENT (DRUM)	4	SHEET 1	-	-
ASM	1	4	CCO LOWER DUNNAGE		SHEET 2	-	-
PRT	2	5	DUNNAGE END PLATE		SHEET 2	PLYWOOD	APA EXTERIOR GRADE C-C
PRT	3	6	DUNNAGE RING PLATE		SHEET 2	PLYWOOD	APA EXTERIOR GRADE C-C
ASM	2	7	FOLDING LIFT HANDLE / INSTALLATION HARDWARE	5	1647A31	STEEL	ZINC PLATED
PRT	12	8	PHILLIPS FLAT HEAD WOOD SCREW, #12 X 3-1/2 LG.		SHEET 2	STEEL	ZINC PLATED
BLK	AR	9	WATERPROOF WOOD GLUE	6	TITEBOND III	POLYMER	ANSI/HPVA TYPE I
ASM	1	10	CRITICALITY CONTROL CONTAINER		SHEET 2	-	-
PRT	1	11	SLIP-ON FLANGE, CLASS 150, NPS 6		SHEET 2	STAINLESS STEEL	ASTM A182, GRADE F304/F304L
PRT	1	12	PIPE, NPS 6, SCH 40		SHEET 2	STAINLESS STEEL	ASTM A312, GRADE TP304/TP304L
PRT	1	13	BLIND FLANGE (BASE), CLASS 150, NPS 6		SHEET 2	STAINLESS STEEL	ASTM A182, GRADE F304/F304L
PRT	1	14	RING GASKET, NPS 6, (6 5/8 ID X 8 3/4 OD X 1/16 THK.	7	DURLON 8500	ARAMID-INORGANIC/NBR	ASTM F104
PRT	1	15	BLIND FLANGE (LID), CLASS 150, NPS 6		SHEET 2	STAINLESS STEEL	ASTM A182, GRADE F304/F304L
PRT	8	16	HEAVY HEX HD BOLT, 3/4-10UNC-2A X 2-1/2 LG.		SHEET 2	STAINLESS STEEL	ASTM A193, GRADE B8, CLASS 1
PRT	1	17	HOIST RING, 2000 LB. WORKING LOAD LIMIT	5	29505T21	ALLOY STEEL	-
PRT	1	18	FILTER VENT (CCC)	4	SHEET 2	-	-
BLK	AR	19	ANAEROBIC THREAD LOCKER/SEALANT		LOCTITE 243	-	-
ASM	1	20	CCO UPPER DUNNAGE		SHEET 2	-	-
PRT	2	21	DUNNAGE END PLATE		SHEET 2	PLYWOOD	APA EXTERIOR GRADE C-C
PRT	3	22	DUNNAGE RING PLATE		SHEET 2	PLYWOOD	APA EXTERIOR GRADE C-C
ASM	2	23	FOLDING LIFT HANDLE / INSTALLATION HARDWARE	5	1647A31	STEEL	ZINC PLATED
PRT	12	24	PHILLIPS FLAT HEAD WOOD SCREW, #12 X 3-1/2 LG.		SHEET 2	STEEL	ZINC PLATED
BLK	AR	25	WATERPROOF WOOD GLUE	6	TITEBOND III	POLYMER	ANSI/HPVA TYPE I

REV	DESCRIPTION	DATE	APPROVED
3	REVISION		
2	REVISION	8/21/13	TS
0	FIRST SUBMITTAL	8/9/10	TS

SIZE	SYSTEM	DWG NO	REV
D	PT01	CCO-DWG-0001	3

SCALE 1:4 | CAD SOURCE Pro/ENGINEER | SHEET 1 OF 2

Figure C-1. Criticality Control Overpack Drawing, Sheet 1 (McGonagill 2015)

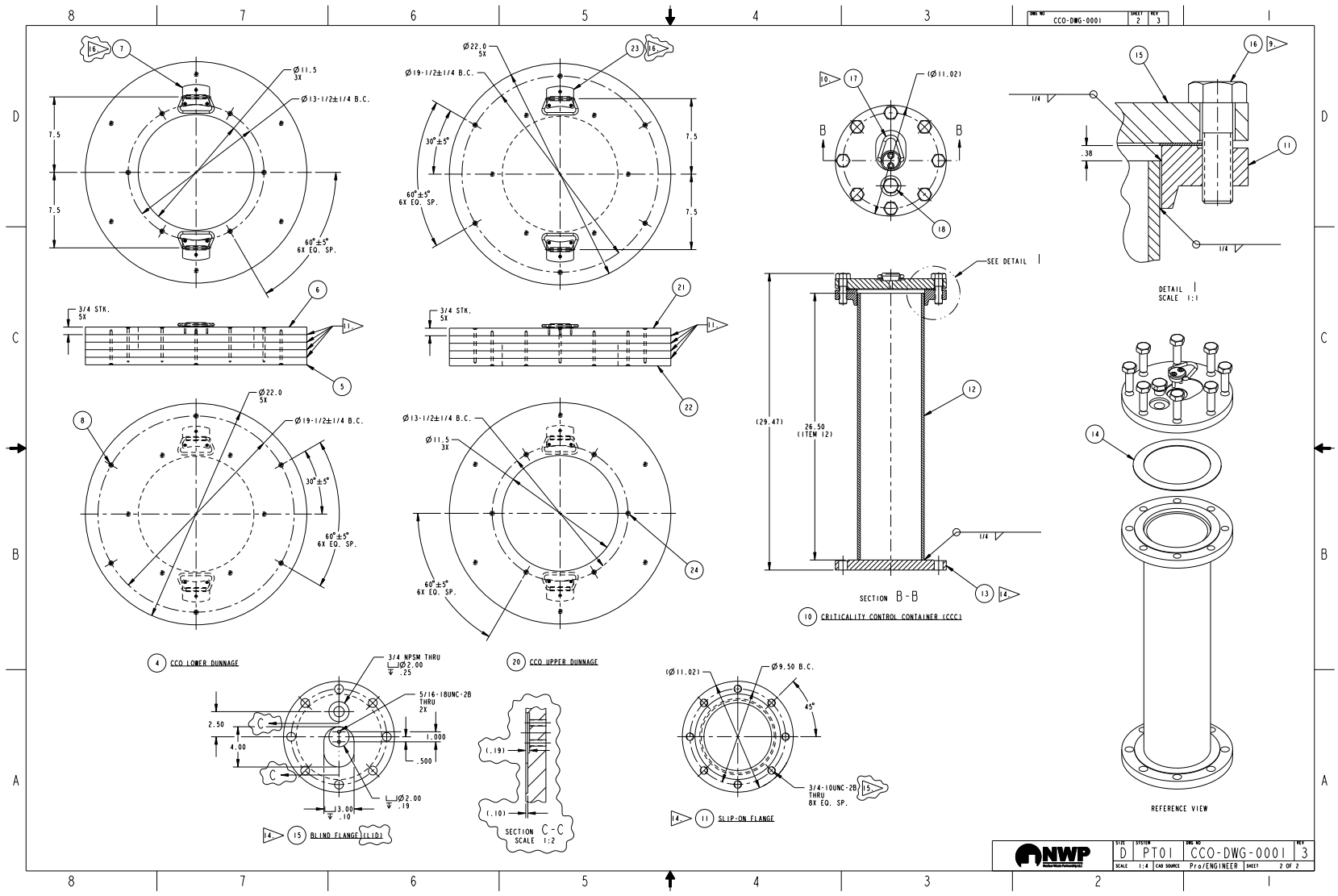
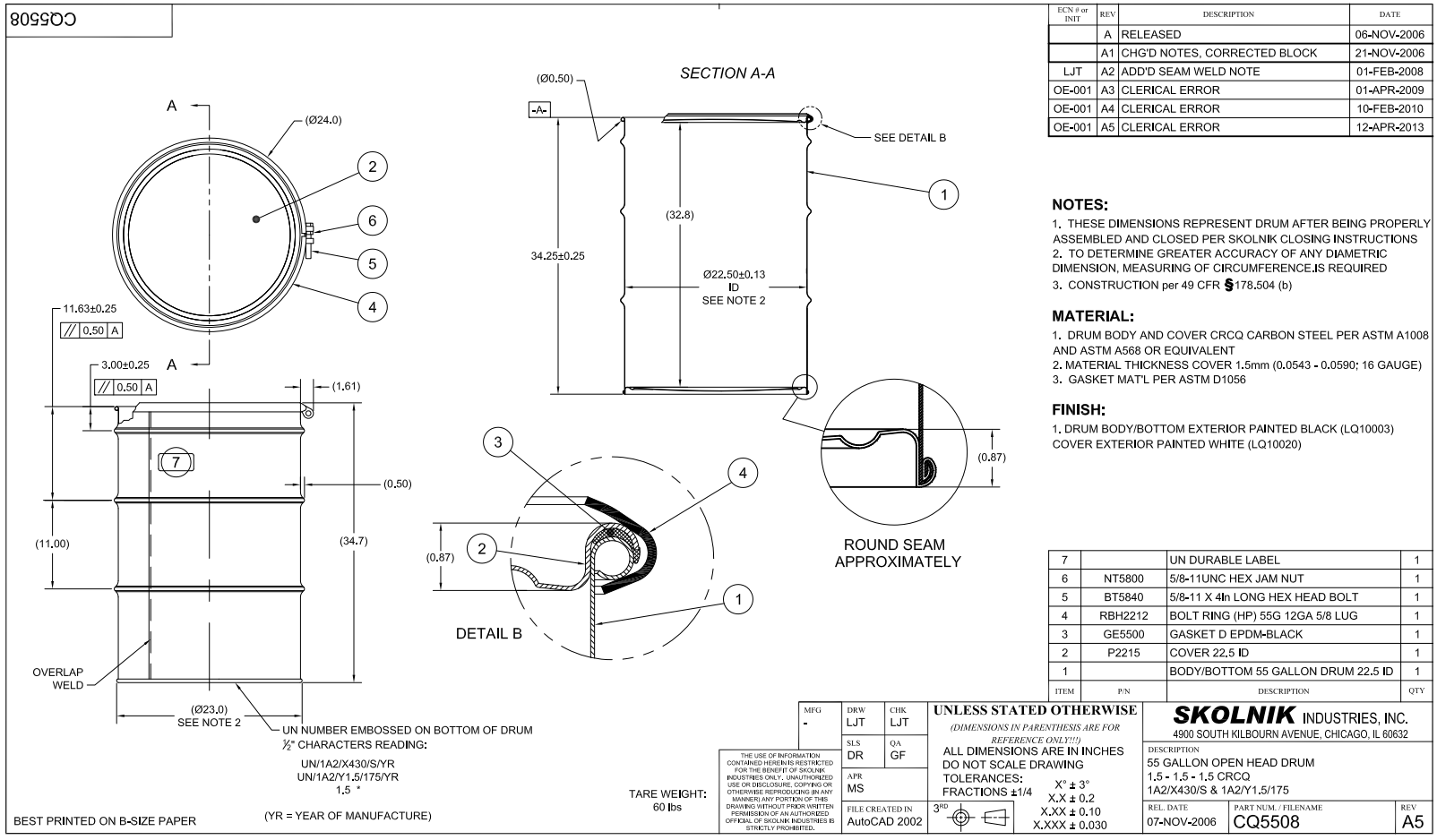


Figure C-2. Criticality Control Overpack Drawing, Sheet 2 (McGonagill 2015)



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	NT5840	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

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

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.

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 C-3. 55-Gallon Drum Detail Drawing (Skolnik Industries 2018)

C.2. Mechanical Response of the Carbon Steel

The mechanical response of the carbon steel was characterized as part of the investigation following the 2014 radiological release at the WIPP (Smith 2014). Several experimental and analysis details are missing in Smith (2014), but discussions with Salzbrenner and Smith (2018) established the following information.

C.2.1. Experimental Setup

A typical 55-gallon drum used for disposal at the WIPP was obtained from Los Alamos National Laboratories and transported to Sandia in Albuquerque. Six specimens were cut from the drum lid, but only drum lid sample 1 (DL1), sample 3 (DL3), and sample 5 (DL5) were tested*. As shown in Fig. C-4a, each specimen's length was oriented approximately radially, but each specimen was located at different circumferential locations around the lid. The nominal specimen geometry conformed to the ASTM-E8 standard, as shown in Fig. C-4b, with initial width W, thickness M, and cross-sectional area $A = W\,M$. The three specimens were uniaxially pulled to failure in a servohydraulic load frame, where the actuator displacement δ was controlled at 0.0254 mm/s while measuring the axial load F. A laser extensometer measured the displacement δ_e between two laser tags affixed to the specimen at a distance of L_e apart. Table C-1 lists the specimens and important measured dimensions (w_f and m_f are defined in Appendix C.2.2).

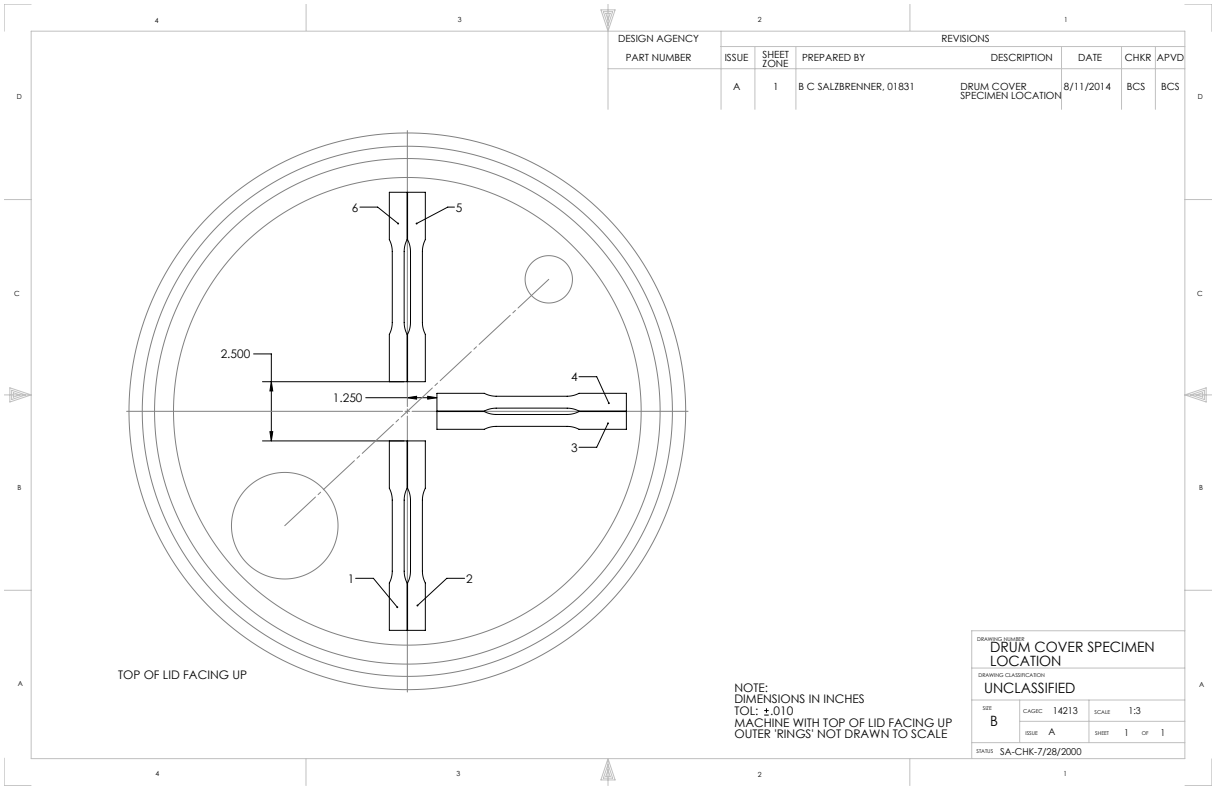
Table C-1. Carbon Steel Specimen Dimensions

Specimen	L_e (mm)	W (mm)	M (mm)	w_f (mm)	m_f (mm)
DL1	47.46	12.66	1.486	7.65	0.478
DL3	52.60	12.62	1.486	7.45	0.518
DL5	53.28	12.64	1.486	7.62	0.452

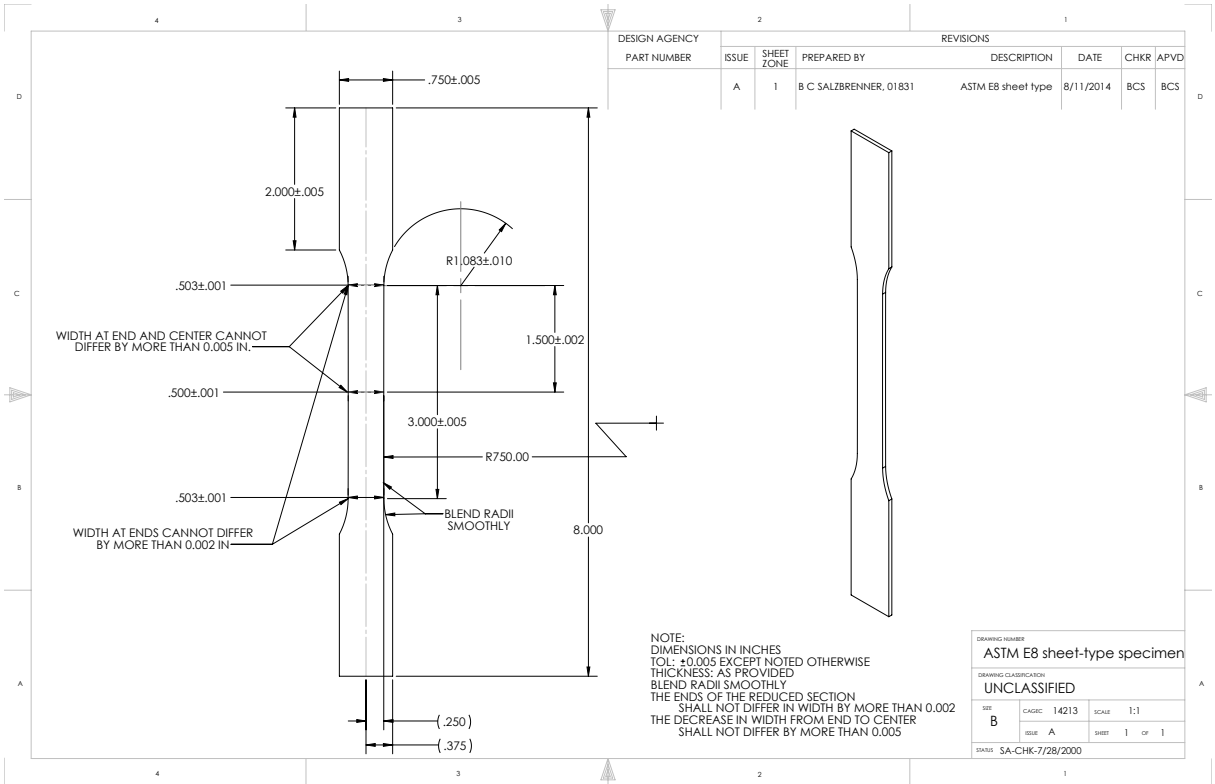
C.2.2. Experimental Results

Figure C-5a depicts the mechanical responses of the three specimens, plotted in terms of engineering stress F/A and engineering strain δ_e/L_e. All three specimens exhibit a small stress drop after yield ($\delta_e/L_e \approx 0.3$ %) and a short period of low work hardening until $\delta_e/L_e \approx 2$ %. The engineering strain histories in Fig. C-5b seem to exhibit higher strain rates for $\delta_e/L_e \approx 0.3$ % to 2 % and slower strain rates before/after. This anomalous behavior is probably due to Lüders bands (Kyriakides and Miller 2000), but further investigation would be needed to confirm. After $\delta_e/L_e = 2$ %, the material work hardens to a ultimate tensile engineering strength of about 335 MPa at $\delta_e/L_e \approx 20$ %, the load drops as the specimen develops a neck, and the specimen fails

*Smith (2014) lists the specimens tested as DL1, DL2, and DL3, but correspondence with Salzbrenner and Smith (2018) revealed that the specimens were actually DL1, DL3 and DL5.



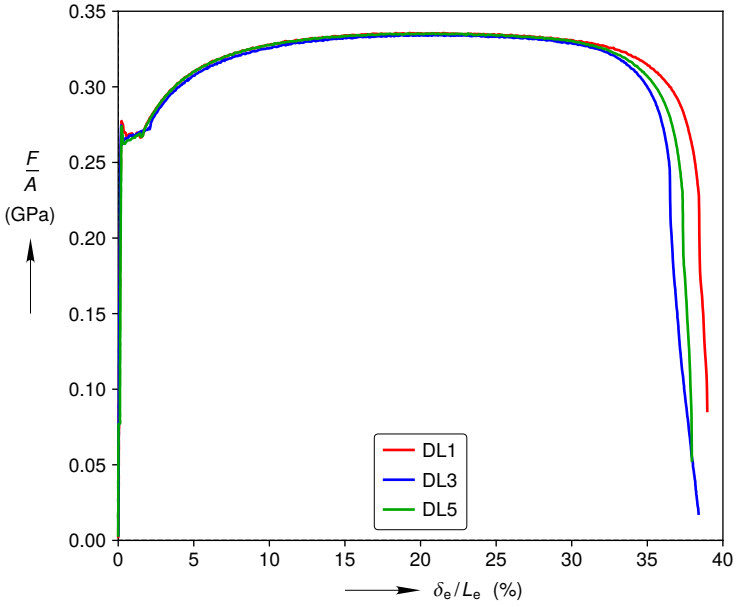
(a) Specimen Locations in the Drum Lid



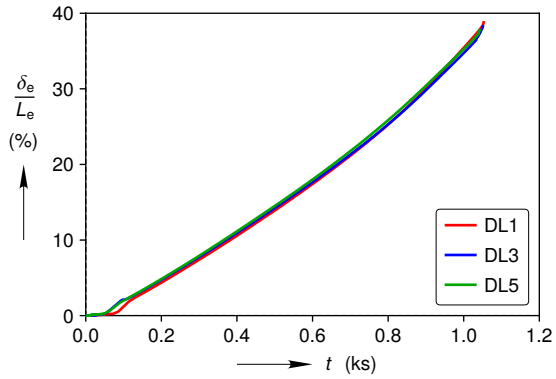
(b) Specimen Dimensions

Figure C-4. Carbon Steel Specimen Locations and Dimensions (Salzbrenner and Smith 2018)

at $\delta_e/L_e \approx 38$ %. After the tests, the specimen width w_f and thickness m_f at the fracture site were measured, as depicted in Fig. C-6, in order to calculate the fracture area $a_f = w_f m_f$.

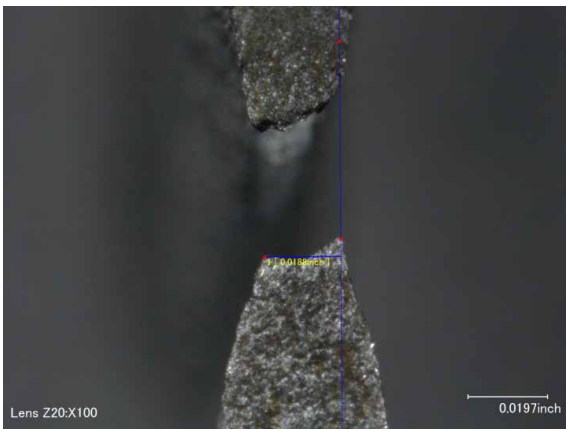


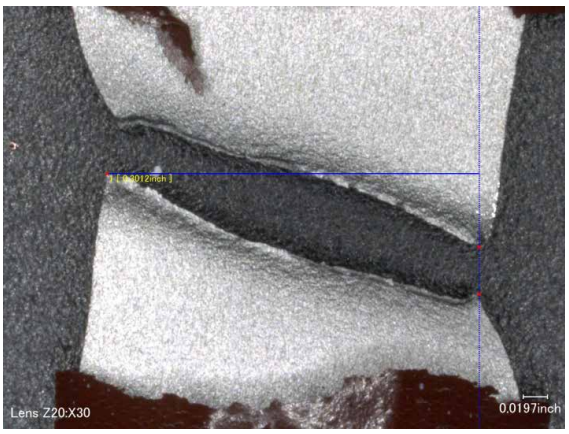
(a) Engineering stress–strain curves



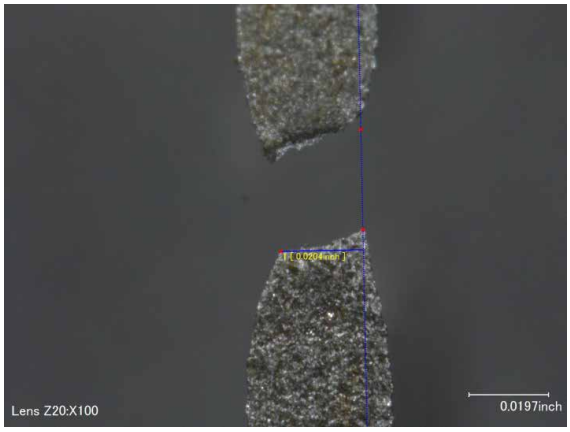
(b) Engineering strain histories

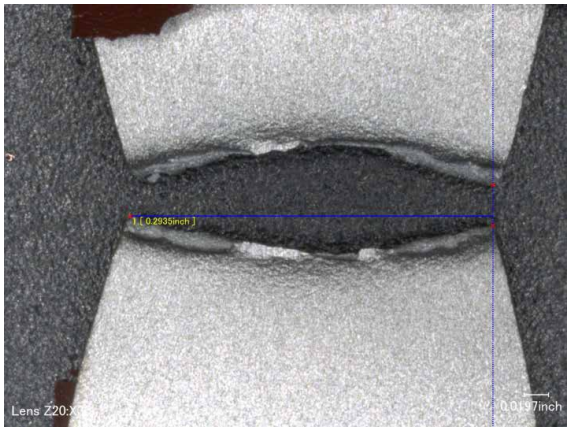
Figure C-5. Carbon steel mechanical response curves and engineering strain histories



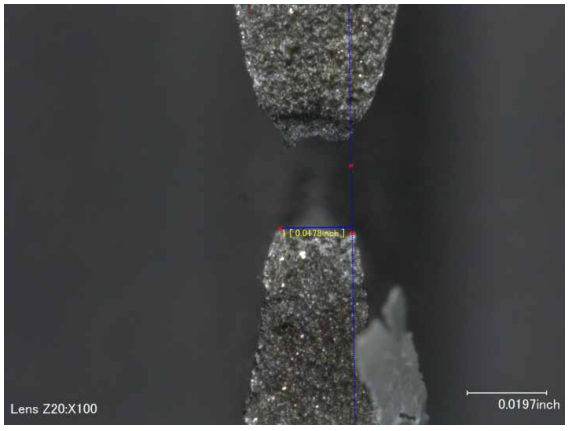


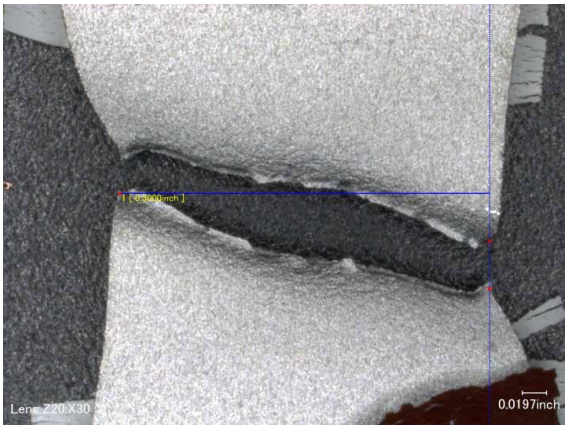
(a) Specimen DL1





(b) Specimen DL3





(c) Specimen DL5

Figure C-6. Photos of drum lid specimen fractures. Left and right images respectively show the thickness and width reductions at the fracture site. (Salzbrenner and Smith 2018)

C.2.3. Hardening Curve Calibration Procedure

The engineering stress-strain curves in Fig. C-5a were converted to a hardening curve by making several standard assumptions. The elastic deformation was assumed to be negligible compared to the plastic deformation for the purpose of calculating the true stress (Cauchy stress) and the strain at failure. Plastic deformation is isochoric, so the current cross-sectional area a and current differential length $\mathrm{d}l$ can be related to their reference counterparts A and $\mathrm{d}L$ by $A\,\mathrm{d}L = a\,\mathrm{d}l$. Deformations are affine prior to necking, which means the isochoric relationship can be rewritten as $A\,L_\mathrm{e} = a\,l_\mathrm{e}$, where $l_\mathrm{e} = \delta_\mathrm{e} + L_\mathrm{e}$. This isochoric relationship allowed the hardening curve to be calculated from the onset of yield to the ultimate tensile engineering stress using the following procedure.

1. The stress strain curve was forced to monotonically increase by smoothing over short duration stress peaks and valleys. (This process effectively ignores the possibility of Lüders bands during $\delta_\mathrm{e}/L_\mathrm{e} \approx 0.3$ % to 2 %.)
2. The true strain (log strain) was computed as $\varepsilon = \ln(\delta_\mathrm{e}/L_\mathrm{e} + 1)$.
3. The hardening was calculated as $H = \sigma = F/a = (F/A)\,(l_\mathrm{e}/L_\mathrm{e}) = (F/A)\exp(\varepsilon)$.
4. The equivalent plastic strain was computed as $\bar{\varepsilon}^\mathrm{pl} = \varepsilon - \sigma/E$, where E is Young's modulus of elasticity.

After the ultimate tensile engineering stress, the specimen develops a neck, which makes precise determination of the hardening curve a non-trivial task and many methods exist. Here, the hardening curve was simply assumed to be linear between the equivalent plastic strain $\bar{\varepsilon}^\mathrm{pl}_\mathrm{uts}$ corresponding to the ultimate tensile engineering stress and the equivalent plastic strain $\bar{\varepsilon}^\mathrm{pl}_\mathrm{f}$ corresponding to failure. The linear slope was determined by the true stress and strain at failure, which were estabilished from the following calculations:

1. The total true strain at failure was calculated as $\varepsilon_\mathrm{f} = \ln(A/a_\mathrm{f})$. (Recall that a_f was the measured fracture area.)
2. The hardening at failure was computed as $H(\bar{\varepsilon}^\mathrm{pl}_\mathrm{f}) = \sigma_\mathrm{f} = F/a_\mathrm{f}$.
3. The equivalent plastic strain at failure was calculated as $\bar{\varepsilon}^\mathrm{pl}_\mathrm{f} = \varepsilon_\mathrm{f} - \sigma_\mathrm{f}/E$.

The three hardening curves from the three specimens were then merged into one by averaging the H values for a given $\bar{\varepsilon}^\mathrm{pl}$ to create the curve shown in Fig. 2-6.

APPENDIX D. Simulation Files

D.1. Comma Separated Variable Files

D.1.1. Stratigraphy

```
material, bot_elev_wrt_G, clay, name, comments
arg_halite, -1000.0, none, none, this depth was made up and assumed to be all arg_halite
polyhalite, -54.19, none, MB 141, none,
arg_halite, -49.99, none, none, none
anhydrite, -30.60, A, MB 140, none
arg_halite, -26.21, none, none, none
anhydrite, -16.41, B, anhydrite c, none
arg_halite, -16.33, none, none, none
arg_halite, -11.37, D, none, none
anhydrite, -8.63, E, MB 139, none
arg_halite, -7.77, none, none, none
arg_halite, -3.72, none, none, none
arg_halite, -2.90, F, none, none
halite, 0.0, G, none, none
anhydrite, 2.10, H, anhydrite a, none
halite, 2.31, none, none, none
arg_halite, 4.27, I, none, none
arg_halite, 6.71, J, none, none
arg_halite, 7.77, none, none, none
anhydrite, 9.16, K, MB 138, none
arg_halite, 9.35, none, none, none
arg_halite, 10.67, none, none, none
arg_halite, 13.58, L, none, none
anhydrite, 28.3, M, MB 136, none
arg_halite, 31.86, none, none, none
anhydrite, 49.38, none, MB 134, none
arg_halite, 52.87, none, none, this was just assumed to be all arg_halite
```

D.2. Cubit Journal Files

D.2.1. 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 numpy as np
import math
import re

mesh_name = 'disposal_room-sharp_corners'

#Initialize dictionary to store geometry values
#(All dimensions are in meters.)
```

```
g = dict()

#Define whether to split each element into 4 elements
split_elements = False
#Define the clay seam slip definition
seam_slip_defn = 'all'
#Define whether to create blocks for the layer of elements on the room surface
room_surf_blocks = False
#Define whether to assume symmetry across the X=0 plane
x_sym = True
#Define how to determine thickness
thickness_determined_by = 'num_seven_packs'
#Define stratigraphy
strat_defn = 'SAND88_2948'

#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.)
#(Then in SAND2007-3334 they moved it again to avoid ground control issues.  The room top
#elevation is 0.0, and the bottom is at -3.96 m.)
g['room_top_elev'] = 0.0
g['room_bot_elev'] = -3.96
g['room_width'] = 5.03 * 2.0
#(The cutting head radius used in room D was 18.0 * 25.4 / 1000.0.  I do not
#know if the disposal rooms used the same cutting head as Room D, but it seems like a
#safe bet.  Here, however, we will use a sharp room radius to avoid element inversions at
#the corner as the room surfaces come together.)
g['room_fillet_radius'] = 0.00001 #18.0 * 25.4 / 1000.0
#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
if thickness_determined_by == 'num_seven_packs':
    #The canister diameter, including the ribs and the bolt ring on the 55 gallon drum, is 24.0 inches,
    #but if you neglect the ribs, then it is 22.5 inches + 1.5 * 2 mm
    can_dia =  22.5 * 25.4 / 1000.0 + 1.5 * 2 / 1000.0
    #Place a small gap between the canisters
    can_gap = 5.0 / 1000.0
    seven_pack_depth = (can_dia + can_gap) + (can_dia + can_gap) * math.cos(math.pi/6) * 2.0
    num_seven_packs = 1
    #An extra half canister had to be added to both sides to avoid slicing through canisters
    g['thickness'] = seven_pack_depth * num_seven_packs + can_dia

#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
#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

if thickness_determined_by == 'element_size':
    g['thickness'] = H_xz

#Define stratigraphy
if strat_defn == 'SAND88_2948':
    raw = np.genfromtxt(strat_defn + '-stratigraphy.csv', delimiter = ', ', names = True, dtype = None)
elif strat_defn == 'SAND88_2948-ERDA_9':
```

```
    #These elevations come from SAND88-2948 and  "Geologic Data for Borehole ERDA-9,
    #Eddy County, New Mexico" by C. L. Jones.  Only Cowden, Union, and marker beds whose
    #thickness is greater than 1 meter were used from the ERDA 9 data.
    raw = np.genfromtxt(strat_defn + '-stratigraphy-v1.csv', delimiter = ', ', names = True, dtype = None)
elif strat_defn == 'SAND97_0795':
    raw = np.genfromtxt(strat_defn + '-stratigraphy.csv', delimiter = ', ', names = True, dtype = None)
else:
    raise IOError('ERROR: did not recognize stratigraphy definition')
layers = dict(material = raw['material'], bot_elev_wrt_G = raw['bot_elev_wrt_G'], clay = raw['clay'])

if seam_slip_defn == 'SAND88_2948':
    #In SAND2012-7525 and SAND88-2948 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_names = ['A', 'B', 'M']
    for no_clay_name in no_clay_names:
        ndx = np.where(layers['clay'] == no_clay_name)[0][0]
        layers['clay'][ndx] = 'none'
elif seam_slip_defn == 'all':
    print("All clay seams active")
elif seam_slip_defn == 'none':
    #Switch all clay seams to non-slipping interfaces, if desired.
    layers['clay'][:] = 'none'
else:
    raise IOError('ERROR: did not recognize clay seam slip definition')

#Calculate the top elevation values
layers['top_elev_wrt_G'] = np.hstack([layers['bot_elev_wrt_G'][1:], g['top_elev']])

#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 curve arc vertex 11 10 radius %r normal 0 -1 0' %(g['room_fillet_radius']))
cubit.cmd('create curve arc vertex 13 12 radius %r normal 0 -1 0' %(g['room_fillet_radius']))
cubit.cmd('create surface curve 13 14 15 16 17 18')
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 bot, clay in zip(layers['bot_elev_wrt_G'], layers['clay']):
    if bot > g['bot_elev'] and bot < g['top_elev']:
        if clay == 'none':
            cubit.cmd('webcut volume all with plane zplane offset %r imprint merge' %(bot))
        else:
            cubit.cmd('webcut volume all with plane zplane offset %r noimprint nomerge' %(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')

if not x_sym:
    #Mirror and merge
    cubit.cmd('volume all copy reflect x')
    cubit.cmd('merge all')

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

#Find which volumes pertain to which materials
material_names = []
vol_IDs = []
material_dict = dict()
for matl, vols in zip(layers['material'], layers['volumes']):
    match = re.match(r'(\w+)', matl)
    material_names.append(match.group(1))
    vol_IDs.append(vols)
#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 clay, bot, vols in zip(layers['clay'], layers['bot_elev_wrt_G'], layers['volumes']):
    if clay != 'none':
        merged_surfaces = cubit.parse_cubit_list('surface', \
            'with z_coord >= ' + str(bot-tol) \
            + ' and with z_coord <= ' + str(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(vols)[1:-1] \
            + ' with z_coord >= ' + str(bot - tol) \
            + ' and with z_coord <= ' + str(bot + tol))
        #Get the slave surfaces by subtracting the master surfaces from all the
        #surfaces at the clay seam
        surfaces = cubit.parse_cubit_list('surface', \
            'with z_coord >= ' + str(bot - tol) \
            + ' and with z_coord <= ' + str(bot + tol))
        slave_surfaces = list(set(surfaces) - set(master_surfaces))
        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_' + clay + '_master\'')
        i = i + 1
        cubit.cmd('sideset ' + str(i) + ' surface ' + str(slave_surfaces)[1:-1])
        cubit.cmd('sideset ' + str(i) + ' name ' \
            + '\'SS_clay_' +  clay + '_slave\'')
        #Print the results for troubleshooting purposes
        print("Slipping clay seam " + clay + " created at elevation = " + str(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(-(left_surf_x - 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()
if thickness_determined_by == 'num_seven_packs':
    base_file_name = mesh_name + '-%d_cco_7pack_deep' %(num_seven_packs)
elif thickness_determined_by == 'element_size':
    base_file_name = mesh_name + '-1_element_deep'
export_path = os.path.join(pwd, base_file_name + '-%d_elements.g' %(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, base_file_name + '-room_surf-%d_elements.g' %(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, base_file_name + '-geom_values.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)
```

D.2.2. Container

```
# This journal file creates a simplified represertnation of a :
# Skolnik Industries 55 Gallon Open Head Drum (Part No. CQ5508)
#
# Created By: John Wilkes
#
# Last Modified: {Version = "2017_11_28"}
#
# 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)
#
# Project: Criticality of CCO storage at WIPP Waste Disposal/Room D
#
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
#                         #
###   ====================    GEOMETRY     ====================   ###
#
##   ~~~~~~~~~~~~~~~~~~~    55-Gallon Drum   ~~~~~~~~~~~~~~~~~~~   ##
# 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"
#
##   ~~~~~~~~~~~~~~~~~~~    Lower Dunnage    ~~~~~~~~~~~~~~~~~~~   ##
# Definitions of Dunnage/Plywood Geometry Variables
# Height of One Plywood Stack: {w_h = 3/4 * in_m} m
# Inside Diameter: {w_id = 11.50 * in_m} m
# Outside Diameter: {w_od = 22.00 * in_m} m
# Height of Exterior 2 Stack of Plywood: {w_hes = 2 * w_h} m
# Height of Interior Circumferential 3 Stack: {w_his = 3 * w_h} m
#
### ------------- 2 Plywood Stack Base ------------- ###
Create Cylinder Height {w_hes} Radius {w_od / 2}
Surface {Id("Surface")} Rename "S_W_L_e"
Volume {Id("Volume")} Rename "V_W_L_e"
Body {Id("Body")} Rename "B_W_L_e"
Body with Name "B_W_L_e" Move Z {d_hrs + d_t + (w_hes / 2)}
#
### ------------- 3 Plywood Stack Wall ------------- ###
Create Cylinder Height {w_his} Radius {w_od / 2}
Surface {Id("Surface")} Rename "S_W_L_i"
Volume {Id("Volume")} Rename "V_W_L_i"
Body {Id("Body")} Rename "B_W_L_i"
Create Cylinder Height {w_his} Radius {w_id / 2}
Volume {Id("Volume")} Rename "V_W_L_i_Cut"
Subtract Volume with Name "V_W_L_i_Cut" from Volume with Name "V_W_L_i"
Body with Name "B_W_L_i" Move Z {d_hrs + d_t + w_hes + (w_his / 2)}
#
##   ~~~~~~~~~~~~~~~~~~~    Upper Dunnage    ~~~~~~~~~~~~~~~~~~~   ##
### ------------- 2 Plywood Stack Top ------------- ###
Create Cylinder Height {w_hes} Radius {w_od / 2}
Surface {Id("Surface")} Rename "S_W_U_e"
Volume {Id("Volume")} Rename "V_W_U_e"
Body {Id("Body")} Rename "B_W_U_e"
Body with Name "B_W_U_e" Move Z {d_h - d_hrs - (w_hes / 2.0)}
#
### ------------- 3 Plywood Stack Wall ------------- ###
```

```
Create Cylinder Height {w_his} Radius {w_od / 2}
Surface {Id("Surface")} Rename "S_W_U_i"
Volume {Id("Volume")} Rename "V_W_U_i"
Body {Id("Body")} Rename "B_W_U_i"
Create Cylinder Height {w_his} Radius {w_id / 2}
Volume {Id("Volume")} Rename "V_W_U_i_Cut"
Subtract Volume with Name "V_W_U_i_Cut" from Volume with Name "V_W_U_i"
Body with Name "B_W_U_i" Move Z {d_h - d_hrs - w_hes - (w_his / 2.0)}
#
##   ~~~~~~~~~~~~~~~~~~~    Base Flange    ~~~~~~~~~~~~~~~~~~~   ##
# Definitions of Flange Variables & Pipe Variable
# Diameter of (All) Flanges: {f_d = 11.02 * in_m} m
# Approximate Thickness of (All) Flanges: {f_t = 1.00 * in_m} m
# Diameter of Pipe: {p_d = (6.625 * in_m) - p_t} m
# Diameter of Bolt Cirlde: {bc_d = 9.5 * in_m} m
# Bolt Diameter: {b_d = 0.75 * in_m} m
# Bolt Head Diameter: {bh_d = (1.443 + 1.383) / 2.0 * in_m} m
# Bolt Head Thickness: {bh_t = 0.5 * in_m} m
Create Cylinder Height {f_t} Radius {p_d / 2}
Curve {Id("Curve")} Rename "C_F_B_i"
Surface {Id("Surface")} Rename "S_F_B_i"
Volume {Id("Volume")} Rename "V_F_B_i"
Body {Id("Body")} Rename "B_F_B_i"
Body with Name "B_F_B_i" Move Z {d_hrs + d_t + w_hes + (f_t / 2)}
#
Create Cylinder Height {f_t} Radius {f_d / 2}
Curve {Id("Curve")} Rename "C_F_B_o"
Surface {Id("Surface")} Rename "S_F_B_o"
Volume {Id("Volume")} Rename "V_F_B_o"
Body {Id("Body")} Rename "B_F_B_o"
Create Cylinder Height {f_t} Radius {p_d / 2}
Volume {Id("Volume")} Rename "V_F_B_Cut"
Subtract Volume with Name "V_F_B_Cut" from Volume with Name "V_F_B_o"
Body with Name "B_F_B_o" Move Z {d_hrs + d_t + w_hes + (f_t / 2)}
##   ~~~~~~~~~~~~~~~~~~~~~~    Pipe     ~~~~~~~~~~~~~~~~~~~~~~   ##
# Definitions of Pipe Variables
# Length of Pipe: {p_l = 26.50 * in_m} m
# Thickness of Pipe Wall: {p_t = 0.280 * in_m} m
# Modified Gap from Top of Pipe to Lid Flange : {p_f_g = 0.38 * in_m}
Create Surface Circle Radius {p_d / 2}
Surface {Id("Surface")} Rename "S_Ptmp"
# Elevation at Top of Pipe: {p_top = d_hrs + d_t + w_hes + f_t + p_l}
Surface with Name "S_Ptmp" Move Z {p_top}
Sweep Curve in Surface with Name "S_Ptmp" Vector 0 0 { - (p_l + f_t)}
Curve {Id("Curve")} Rename "C_P_W"
Surface {Id("Surface")} Rename "S_P_W"
Volume {Id("Volume")} Rename "V_P_W"
Body {Id("Body")} Rename "B_P_W"
Delete Surface with Name "S_Ptmp"
#
##   ~~~~~~~~~~~~~~~~~~~    Lid Flange     ~~~~~~~~~~~~~~~~~~~   ##
# Inner Portion
Body with Name "B_F_B_i" Copy Move Z {f_t + p_l + p_f_g}
Surface {Id("Surface")} Rename "S_F_L_i"
Volume {Id("Volume")} Rename "V_F_L_i"
Body {Id("Body")} Rename "B_F_L_i"
# Outer Portion
Body with Name "B_F_B_o" Copy Move Z {f_t + p_l + p_f_g }
Curve {Id("Curve")} Rename "C_F_L_o"
Surface {Id("Surface")} Rename "S_F_L_o"
Volume {Id("Volume")} Rename "V_F_L_o"
Body {Id("Body")} Rename "B_F_L_o"
# Bolt Heads
create cylinder height {bh_t} radius {max(bc_d/2.0 + bh_d/2.0,f_d/2.0)}
volume {Id("volume")} rename "Bolt_Heads"
create cylinder height {bh_t} radius {bc_d/2.0 - bh_d/2.0}
volume {Id("volume")} rename "Discard"
subtract volume with name "Discard*" from volume with name "Bolt_Heads"
```

```
volume with name "Bolt_Heads*" move z {bh_t/2.0 + d_hrs + d_t + w_hes + f_t + f_t + p_l + p_f_g}
#
##   ~~~~~~~~~~~~~~~~~~~    Slip Flange    ~~~~~~~~~~~~~~~~~~~   ##
Body with Name "B_F_L_o" Copy Move Z {- f_t}
Curve {Id("Curve")} Rename "C_F_S"
Surface {Id("Surface")} Rename "S_F_S"
Volume {Id("Volume")} Rename "V_F_S"
Body {Id("Body")} Rename "B_F_S"
#
##   ~~~~~~~~~~~~~~~~~~    Slip Flange Hub    ~~~~~~~~~~~~~~~~~~   ##
# Definitions of Slip Flange Hub Variables
# Top Diameter of Slip Flange Hub: {f_tdh = p_d + 0.47 * in_m} m
# Bottom Diameter of Slip Flange Hub: {f_bdh = p_d + 0.32 * in_m} m
# Height of Slip Flange Hub: {f_hh = 0.56 * in_m} m

# Create Frustum Height {f_hh} Major Radius {f_bdh / 2} Top {f_tdh / 2}
# Curve {Id("Curve")} Rename "C_F_S_H"
# Surface {Id("Surface")} Rename "S_F_S_H"
# Volume {Id("Volume")} Rename "V_F_S_H"
# Body {Id("Body")} Rename "B_F_S_H"
# Create Cylinder Height {f_hh} Radius {p_d / 2}
# Volume {Id("Volume")} Rename "V_F_S_H_Cut"
# Subtract Volume with Name "V_F_S_H_Cut" from Volume with Name "V_F_S_H"
# Body with Name "B_F_S_H" Move Z {p_top + p_f_g - f_t - (f_hh / 2)}

#
#
###   ==================   QUARTER SYMMETRY   ==================   ###
#
Webcut Volume All with Plane yplane center 0
Webcut Volume All with Plane xplane center 0
#
Delete Volume with Name "*@*"
#
###    =========    WEBCUTS FOR MESH UNIFORMITY      =========    ###
## --  Dunnage Intersections -- ##
Webcut Volume with Name "V_W_*_e" with Volume with Cylinder Radius {w_id / 2} Axis Z Center 0 0 0
#
## --  Pipe and Flange Intersections -- ##
# - Cut Pipe Wall at Top of Base Flange - #
# Top of Base Flange: {fb_top = d_hrs + d_t + w_hes + f_t}
Webcut Volume with Name "V_P_W" with Plane zplane Offset {fb_top}
#
# - Cut Pipe Wall at Bottom of Slip Flange Hub - #
# Define vertical line along outer Slip Flange Hub edge as variable for meshing density
Curve with x_coord > {0.99 * (f_bdh / 2)} in Volume with Name "V_F_S_H" Rename "C_F_S_H_vert"
# btm of slip flange hub: {fsh_btm = p_top + p_f_g - f_t - f_hh}
Webcut Volume with Name "V_P_W*" with Plane zplane offset {fsh_btm}
#
# - Cut Pipe Wall at Top of Slip Flange Hub/Bottom of Slip Flange - #
# Btm of Slip Flange: {fs_btm = fsh_btm + f_hh}
Webcut Volume with Name "V_P_W*" with Plane zplane Offset {fs_btm}
#
# - Cut Slip Flange at Top of Pipe Wall - #
# Define vertical line along outer Slip Flange edge as variable for meshing density
Curve with x_coord > {0.99 * (f_d / 2)} in Volume with Name "V_F_S" Rename "C_F_S_vert"
Webcut Volume with Name "V_F_S" with Plane zplane Offset {p_top}
#
# - Cut Slip Flange along Exterior Edge of Slip Flange Hub - #
#Webcut Volume with Name "V_F_S*" with Cylinder Radius {f_tdh/2} Axis Z Center 0 0 0
# - Cut Lid Flange along Exterior Edge of Slip Flange Hub - #
#Webcut Volume with Name "V_F_L*" with Cylinder Radius {f_tdh/2} Axis Z Center 0 0 0
#
webcut volume with name "V_F_L*" or with name "V_F_S*" with cylinder radius {bc_d/2.0 - bh_d/2} axis z center 0
↪  0 0
#
#
###    ====================    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*"
#
##   ~~~~~~~~~~~~~~~~~~~    Lower Dunnage    ~~~~~~~~~~~~~~~~~~~   ##
merge volume with name "V_W*"
volume with name "V_W*" size {w_hes/(1 * mesh_factor)}
mesh volume with name "V_W*"
#
##   ------------- Lid Flange  ------------- ###
merge volume with name "V_F*" or with name "Bolt_Heads*"
volume with name "V_F*" or with name "Bolt_Heads*" size {f_t/(1 * mesh_factor)}
mesh volume with name "V_F*" or with name "Bolt_Heads*"
#
##   ------------- Pipe ------------- ###
# Merge All Portions on Pipe with their respective adjoining flanges
Merge Volume with Name "V_P_W*" with Volume with Name "V_F_*"
# Merge All Portions of Pipe
Merge Volume with Name "V_P_W*"
volume with name "V_P*" size {f_t/(1 * mesh_factor)}
# Mesh All Pipe Surfaces
mesh surface in volume with name "V_P*"
#
#
###   ===   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_W*"
Merge Volume with Name "V_F*" or with name "Bolt_Heads*"
Merge Volume with Name "V_P*"
#
#
###  === Create Blocks  ===  ###
block 100 surface in volume with name "V_D_W*"
block 100 name 'cco_drum_wall'
#
block 101 surface in volume with name "V_D_B*"
block 101 name 'cco_drum_base'
#
block 102 surface in volume with name "V_D_L*"
block 102 name 'cco_drum_lid'
#
block 200 volume with name "V_W_L*"
block 200 name 'cco_wood_lower'
#
block 201 volume with name "V_W_U*"
block 201 name 'cco_wood_upper'
#
block 300 surface in volume with name "V_P_W*"
block 300 name 'cco_cv_wall'
#
block 301 volume with name "V_F_B*"
block 301 name 'cco_cv_base'
#
block 302 volume with name "V_F_S*"
block 302 name 'cco_cv_top_flange'
#
block 303 volume with name "V_F_L*" or with name "Bolt_Heads*"
block 303 name 'cco_cv_lid'
#
# Move assembly so that it is centered at 0 0 0
group 'cco_assembly' add volume all
move group with name 'cco_assembly' midpoint location 0.0 0.0 0.0
# Move upper dunage so it is in contact with cv.
```

```
move volume with name 'V_W_U*' z {-(d_h - 2.0*d_hrs - 2.0*w_hes - d_t - p_l - p_f_g - 2.0*f_t - bh_t)}
↪  include_merged

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

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

D.2.3. Container Emplacement

Emplacement Part 1

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

# Define conversion factors
in_m = 0.0254

#Define the output mesh name
mesh_name = 'detailed_cco_row_2017_11_28_part1'

#Define the CCO and disposal room names
cco_mesh_name = 'detailed_cco_2017_11_28'

#Import the CCO
cubit.cmd('import cubit \'' + cco_mesh_name + '.cub\'')

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

vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
for theta in theta_list:
   dx = [R * math.cos(theta), 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 * (cco_g['D'] + gap) * math.cos(math.pi/6.0) + cco_g['D']  + gap # Added gap JLB
g['L_x'] = g['L_y'] / math.cos(math.pi/6.0) # JLB
g['L_z'] = cco_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 "cco_stack_assembly" add volume all')

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

cubit.cmd('move group with name "cco_stack_assembly" midpoint location %r %r %r' %(dx[0], dx[1], dx[2]))


#Assemble into a row of CCOs
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 CCOs
#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]))
#
## Remove volumes outside of symmetry planes
#cubit.cmd('delete volume with y_coord < 0.0')
#cubit.cmd('delete volume with y_coord > %r' %(g['L_y']))
#
#Stack the row of CCOs 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]
for i in i_list:
    dx = [0.0, 0.0, float(i) * cco_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 CCO's so that the assembly is centered at 0 0 0
cubit.cmd('group "cco_row_assembly" add volume all')
cubit.cmd('move group with name "cco_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 "cco_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 "cco_drum_wall"')
#
cubit.cmd('block 101 surface in volume with name "V_D_B*"')
cubit.cmd('block 101 name "cco_drum_base"')
#
cubit.cmd('block 102 surface in volume with name "V_D_L*"')
cubit.cmd('block 102 name "cco_drum_lid"')
#
cubit.cmd('block 200 volume with name "V_W_L*"')
cubit.cmd('block 200 name "cco_wood_lower"')
#
cubit.cmd('block 201 volume with name "V_W_U*"')
cubit.cmd('block 201 name "cco_wood_upper"')
#
cubit.cmd('block 300 surface in volume with name "V_P_W*"')
cubit.cmd('block 300 name "cco_cv_wall"')
#
cubit.cmd('block 301 volume with name "V_F_B*"')
cubit.cmd('block 301 name "cco_cv_base"')
#
cubit.cmd('block 302 volume with name "V_F_S*"')
cubit.cmd('block 302 name "cco_cv_top_flange"')
#
cubit.cmd('block 303 volume with name "V_F_L*" or with name "Bolt_Heads*"')
cubit.cmd('block 303 name "cco_cv_lid"')
```

```
## Create Symmetry Node Sets
#cubit.cmd('nodeset 1000 add node with y_coord > %r in volume with is_sheetbody' %(g['L_y']/2.0 - 0.0005))
#cubit.cmd('nodeset 1000 name "ns_sym_shell_pos_y"')
#
#cubit.cmd('nodeset 1001 add node with y_coord < %r in volume with is_sheetbody' %(-(g['L_y']/2.0 - 0.0005)))
#cubit.cmd('nodeset 1001 name "ns_sym_shell_neg_y"')
#
#cubit.cmd('nodeset 2000 add node with y_coord > %r in volume with not is_sheetbody' %(g['L_y']/2.0 - 0.0005))
#cubit.cmd('nodeset 2000 name "ns_sym_hex_pos_y"')
#
#cubit.cmd('nodeset 2001 add node with y_coord < %r in volume with not is_sheetbody' %(-(g['L_y']/2.0 - 0.0005)))
#cubit.cmd('nodeset 2001 name "ns_sym_hex_neg_y"')
#

#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_2017_11_28_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)
```

Emplacement Part 2

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

# Define conversion factors
in_m = 0.0254

#Define the output mesh name
mesh_name = 'detailed_cco_row_2017_11_28_part2'

#Define the CCO and disposal room names
cco_mesh_name = 'detailed_cco_2017_11_28'

#Import the CCO
cubit.cmd('import cubit \'' + cco_mesh_name + '.cub\'')

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

```
cco_g['L'] = (34.25 - 2.0*0.87) * in_m
R = cco_g['D'] + gap

vol_list = cubit.parse_cubit_list('volume', 'all')
vol_list_str = str(vol_list)[1:-1]
for theta in theta_list:
   dx = [R * math.cos(theta), 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 * (cco_g['D'] + gap) * math.cos(math.pi/6.0) + cco_g['D']  + gap # Added gap JLB
g['L_x'] = g['L_y'] / math.cos(math.pi/6.0) # JLB
g['L_z'] = cco_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 "cco_stack_assembly" add volume all')
cubit.cmd('move group with name "cco_stack_assembly" midpoint location 0.0 0.0 0.0')


#Assemble into a row of CCOs
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 CCOs
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]))

# Remove volumes outside of symmetry planes
print cco_g['D']
print R
#cubit.cmd('delete volume with y_coord < 0.0')
#cubit.cmd('delete volume with y_coord > %r' %(g['L_y']))
cubit.cmd('delete volume with y_coord < %r' %(-cco_g['D']/2))
cubit.cmd('delete volume with y_coord > %r' %(g['L_y']+cco_g['D']/2))


#Stack the row of CCOs 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]
for i in i_list:
    dx = [0.0, 0.0, float(i) * cco_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 CCO's so that the assembly is centered at 0 0 0
cubit.cmd('group "cco_row_assembly" add volume all')
cubit.cmd('move group with name "cco_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 "cco_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 "cco_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 "cco_drum_lid_2"')
cubit.cmd('block 102 renumber 1020')
#
cubit.cmd('block 200 volume with name "V_W_L*"')
cubit.cmd('block 200 name "cco_wood_lower_2"')
cubit.cmd('block 200 renumber 2000')
#
cubit.cmd('block 201 volume with name "V_W_U*"')
cubit.cmd('block 201 name "cco_wood_upper_2"')
cubit.cmd('block 201 renumber 2010')
#
cubit.cmd('block 300 surface in volume with name "V_P_W*"')
cubit.cmd('block 300 name "cco_cv_wall_2"')
cubit.cmd('block 300 renumber 3000')
#
cubit.cmd('block 301 volume with name "V_F_B*"')
cubit.cmd('block 301 name "cco_cv_base_2"')
cubit.cmd('block 301 renumber 3010')
#
cubit.cmd('block 302 volume with name "V_F_S*"')
cubit.cmd('block 302 name "cco_cv_top_flange_2"')
cubit.cmd('block 302 renumber 3020')
#
cubit.cmd('block 303 volume with name "V_F_L*" or with name "Bolt_Heads*"')
cubit.cmd('block 303 name "cco_cv_lid_2"')
cubit.cmd('block 303 renumber 3030')


# Create Symmetry Node Sets
#cubit.cmd('nodeset 1000 add node with y_coord > %r in volume with is_sheetbody' %(g['L_y']/2.0 - 0.0005))
#cubit.cmd('nodeset 1000 add node with y_coord = %r in volume with is_sheetbody' %(g['L_y']/2.0))

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 < %r in volume with is_sheetbody' %(-(g['L_y']/2.0 - 0.0005)))
#cubit.cmd('nodeset 1001 add node with y_coord = %r in volume with is_sheetbody' %(-(g['L_y']/2.0)))

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 > %r in volume with not is_sheetbody' %(g['L_y']/2.0 - 0.0005))
#cubit.cmd('nodeset 2000 add node with y_coord = %r in volume with not is_sheetbody' %(g['L_y']/2.0))

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 < %r in volume with not is_sheetbody' %(-(g['L_y']/2.0 - 0.0005)))
#cubit.cmd('nodeset 2001 add node with y_coord = %r in volume with not is_sheetbody' %(-(g['L_y']/2.0)))

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_2017_11_28_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-1_cco_row_2017_11_28_nocut_thickerRoom_new'

#Define the CCO and disposal room names
cco_name = 'detailed_cco_row'
cco_mesh_name_1 = cco_name + '_2017_11_28_part1'
cco_mesh_name_2 = cco_name + '_2017_11_28_part2'
dr_name = 'disposal_room-with_clay_slip-1_cco_row_deep'
dr_mesh_name = dr_name + '-24_elements_2017_11_28'

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 CCO and disposal room
dr_g = load_pickled_file(dr_name + '-geom_values_2017_11_28.pkl')
cco_g = load_pickled_file(cco_name + '-geom_values_2017_11_28_part1.pkl')

#Import the CCO(s)
#cubit.cmd('import cubit \" + cco_name + '.cub\"')
cubit.cmd('import mesh geometry \'' + cco_mesh_name_1 + '.g\' unique genesis ids')
cubit.cmd('import mesh geometry \'' + cco_mesh_name_2 + '.g\' unique genesis ids')

#Move the CCO to the disposal room
cco_vol_list = cubit.parse_cubit_list('volume', 'all')
cco_vol_list_str = str(cco_vol_list)[1:-1]
cubit.cmd('group "cco_row_assembly" add volume all')
cubit.cmd('move group with name "cco_row_assembly" midpoint location 0.0 0.0 0.0')
```

```
#dx = [0.0, dr_g['thickness']/4.0, dr_g['room_bot_elev']]
dx = [ 0.0, 0.0, dr_g['room_bot_elev'] + cco_g['L_z']/2.0]
cubit.cmd('move volume ' + cco_vol_list_str + ' x %r y %r z %r include_merged' %(dx[0], dx[1], dx[2]))

#Import the disposal room
#cubit.cmd('import cubit \" + dr_mesh_name + '.cub\")
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)
```

D.3. Sierra Input Files

D.3.1. Empty Disposal Room Simulations

Implicit Quasi-statics

```
begin sierra WIPP Isothermal Disposal Room

    title Simulation of WIPP Isothermal Disposal Room

    #---------- Variables ---------

    #Define the time periods
    $-0.01 #s
    $0 #s
    $0 #s
    $3.15576e+10 #s

    #---------- Directions --------

    define direction y with vector 0.0 1.0 0.0
    define direction x with vector 1.0 0.0 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 values
  end definition for function function_constant

    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 / 0.01))"
    end definition for function fluid_density_ramp
```

```
  #---------- 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, except
        #steady-state mechanism 3 was turned off.)
        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            = 0.0 # 1/sec
        a2            = 4.415e16 # 1/sec
        q2/R          = 5123 # K
        n2            = 6.279
        b2            = 0.0 # 1/sec
        sigma_g       = 20570000.0 # Pa
        q             = 0.0
        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_th      = 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, but SAND97-0795 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

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

  #---------- Finite Element Model -----------

  begin finite element model room
    Database name = %B.g
    Database type = exodusII

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

   begin parameters for block b_arg_halite
     material halite
     solid mechanics use model md_viscoplastic
     section = solid_1
```

```
   end parameters for block b_arg_halite

   begin parameters for block b_anhydrite
     material anhydrite
     solid mechanics use model kayenta
     section = solid_1
   end parameters for block b_anhydrite

   begin parameters for block b_polyhalite
     material polyhalite
     solid mechanics use model soil_foam
     section = solid_1
   end parameters for block b_polyhalite

  end finite element model room

begin adagio procedure The_Procedure.1

    begin time control

        begin time stepping block p0
              start time = -0.01
              begin parameters for adagio region SmRegion.1
                  time increment = 5e-4
              end parameters for adagio region SmRegion.1
        end time stepping block p0

        begin time stepping block p1
              start time = 0
              begin parameters for adagio region SmRegion.1
                  time increment = 0.1
              end parameters for adagio region SmRegion.1
        end time stepping block p1

          termination time = 3.15576e+10

      end time control

    #---------- Time Step Control ----------


  begin adagio region SmRegion.1

      use finite element model room

      begin adaptive time stepping time
         #The md_viscoplastic model does not generate a material suggested time step, so
         #the material suggested time step will be infinite, causing the time step to
         #always grow, unless the solver fails to converge.
         method = material
         cutback factor = 0.8
         growth factor = 1.0 during p0
         growth factor  = 1.02 during p1

         maximum multiplier = 1 during p0
         maximum multiplier = 78894000 during p1
         minimum multiplier = 1.e-8
         maximum failure cutbacks = 20
      end adaptive time stepping time

      #--------- Boundary Conditions ---------

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

```
      function = function_constant
    end gravity

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

    begin pressure
      surface = SS_top_surf
      function = function_constant
      scale factor = 13.57E+06
    end pressure

    begin fixed displacement
      node set = NS_bot_surf
      components = z
    end fixed displacement

    begin fixed displacement
      node set = NS_left_surf
      components = x
    end fixed displacement

    begin fixed displacement
      node set = NS_right_surf
      components = x
    end fixed displacement

    begin fixed displacement
      node set = NS_back_surf
      components = y
    end fixed displacement

    begin fixed displacement
      node set = NS_front_surf
      components = y
    end fixed displacement

    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 fluid_pressure

#     --------- CONTACT MODEL --------------

  begin contact definition contact_defn
```

```
enforcement = al
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

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

begin constant friction model clay_seam_model
  friction coefficient = 0.2
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 interaction

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

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

begin interaction clay_G
  master = clay_G_master
```

```
      slave = clay_G_slave
      friction model = clay_seam_model
    end interaction

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

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

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

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

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

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

    begin contact surface room_surf
        surface = SS_room_surf
    end

    begin constant friction model salt_to_salt_friction_model
        friction coefficient = 0.5
    end

      #In February 2020, we realized this command contains a mistake.  It was intended
     #to only specify mu_s = 0.5 between the room surface and each clay seam surface,
     #but it also set mu_c = 0.5 for clay seam F and G.
    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
        interaction behavior = sliding
    end

end contact definition contact_defn

  #--------- 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

#--------- User Output Variable Definitions ----------

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
end user output

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_right_wall_ctr
  compute global right_wall_disp as average of nodal displacement
end

#--------- Results Output ---------

begin results output output_1
  database name = %B-r.e
  database type = exodusII
  At time -0.01, increment = 0.005
  At time 0, increment = 315576000
  nodal variables = displacement as disp
  nodal variables = contact_accumulated_slip
  nodal variables = residual
  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
end results output output_1

#--------- History Output ---------

begin history output output_3
  database name = %B-h.e
  database type = exodusII
  At time -0.01, increment = 0.002
  at time 0 increment = 0.0864 #s
  at time 0.864 increment = 0.864 #s
  at time 8.64 increment = 8.64 #s
```

```
        at time 86.4 increment = 86.4 #s
        at time 864 increment = 864 #s
        at time 8640 increment = 8640 #s
        at time 86400 increment = 86400 #s
        at time 864000 increment = 864000 #s
        at time 31557600 increment = 7889400 #s
        global time
        global wall_clock_time as run_time
        global roof_disp
        global floor_disp
        global right_wall_disp
      end history output output_3

      #--------- Restart ---------

        begin restart data restart_1
            database name = %B.rsout
            overwrite = True
            at wall time 0.0h increment = 24.0h
        end restart data restart_1

      #--------- Solver ---------

      Begin solver

      begin loadstep predictor
        type = scale_factor
        scale factor = 0.0 0.0
      end loadstep predictor

        begin control contact
          level = 1
          target relative residual = 1e-4
          acceptable relative residual = 1e-4
          target residual = 1e-4
          acceptable residual = 1e-4
          maximum iterations = 100
          lagrange adaptive penalty = off
          #developer lagrange target gap = 1.0e-5
        end control contact

        begin cg
          reference = external
          target relative residual = 1.0e-5
          acceptable relative residual = 1.0e30
          target residual = 1.0e-5
          acceptable residual = 1.0e30
          maximum iterations = 100
          iteration print = 1
          line search secant 1e-3
          begin full tangent preconditioner
            linear solver = my_feti
            conditioning = no_check
            small number of iterations = 50
            minimum smoothing iterations = 10
            iteration update = 50
          end
        end cg

      end solver

  end adagio region SmRegion.1

end adagio procedure The_Procedure.1

begin feti equation solver my_feti
      corner algorithm = 14
end
```

```
end sierra WIPP Isothermal Disposal Room
```

Explicit Dynamics

```
begin sierra WIPP Isothermal Disposal Room

    title Simulation of WIPP Isothermal Disposal Room

    #---------- Variables ---------

    #Define the time periods
    ${t_0_0 = -5.0e-1} #s
    ${t_0_1 = 0.0} #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 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 = 0.0}
    ${t_m_1_0 = t_m_0_1} #s
    ${t_m_1_1 = 86400.0 * 365.25 * 1000.0} #s

    #---------- Directions --------

    define direction y with vector 0.0 1.0 0.0
    define direction x with vector 1.0 0.0 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 values
  end definition for function function_constant

    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 / {t_0_1 - t_0_0}))"
    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
            {t_0_0}                     "{s_0_0}"
            {t_0_1 + 1e-10}             "{s_1_0} + {s_ramp} * time"
        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, but SAND97-0795 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

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

  #---------- Finite Element Model -----------

  begin finite element model room
    Database name = %B.g
    Database type = exodusII

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

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

   begin parameters for block b_anhydrite
     material anhydrite
     solid mechanics use model kayenta
     section = solid_1
   end parameters for block b_anhydrite

   begin parameters for block b_polyhalite
     material polyhalite
     solid mechanics use model soil_foam
     section = solid_1
   end parameters for block b_polyhalite

  end finite element model room

begin adagio procedure The_Procedure.1

    begin time control

        begin time stepping block p0
              start time = {t_0_0}
```

```
            begin parameters for presto region SmRegion.1
            end parameters for presto region SmRegion.1
      end time stepping block p0

      begin time stepping block p1
            start time = {t_1_0}
            begin parameters for presto region SmRegion.1
            end parameters for presto region SmRegion.1
      end time stepping block p1

        termination time = {t_1_1}

    end time control

  #---------- Time Step Control ----------


begin presto region SmRegion.1

      begin solution termination
          terminate global mat_time > {t_m_1_1}
      end

    use finite element model room

    #--------- Boundary Conditions ---------

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

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

    begin pressure
      surface = SS_top_surf
      function = function_constant
      scale factor = 13.57E+06
    end pressure

    begin fixed displacement
      node set = NS_bot_surf
      components = z
    end fixed displacement

    begin fixed displacement
      node set = NS_left_surf
      components = x
    end fixed displacement

    begin fixed displacement
      node set = NS_right_surf
      components = x
    end fixed displacement

    begin fixed displacement
      node set = NS_back_surf
      components = y
    end fixed displacement
```

```
      begin fixed displacement
        node set = NS_front_surf
        components = y
      end fixed displacement

      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 fluid_pressure

#      --------- CONTACT MODEL --------------

    begin contact definition contact_defn

        begin enforcement options
            momentum balance iterations = 20
        end

        enforcement = al
        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

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

```
begin constant friction model clay_seam_model
  friction coefficient = 0.2
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 interaction

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

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

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

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

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

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

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

begin interaction clay_L
  master = clay_L_master
  slave = clay_L_slave
```

```
      friction model = clay_seam_model
    end interaction

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

    begin contact surface room_surf
        surface = SS_room_surf
    end

    begin constant friction model salt_to_salt_friction_model
        friction coefficient = 0.5
    end

     #In February 2020, we realized this command contains a mistake.  It was intended
     #to only specify mu_s = 0.5 between the room surface and each clay seam surface,
     #but it also set mu_c = 0.5 for clay seam F and G.
    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
        interaction behavior = sliding
    end

end contact definition contact_defn

  #--------- 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 = {t_0_0}
    end

    begin user variable mat_time
        type = global real length = 1
        global operator = min
        initial value = {t_m_0_0}
    end

    begin user variable vp_rate_scale_factor
```

```
      type = global real length = 1
      global operator = min
      initial value = {s_0_0}
  end

  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.
      ${ndx = 0}
      {Loop(8)}
      compute element vp_rate_scale_factor({ndx += 1}) as function
      ↪  vp_rate_scale_factor_broadcast_function
      {EndLoop}
      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

#--------- User Output Variable Definitions ----------

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
end user output

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_right_wall_ctr
  compute global right_wall_disp as average of nodal displacement
end

#--------- Results Output ---------

begin results output output_1
  database name = %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 as 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
      end results output output_1

      #--------- History Output ---------

      begin history output output_3
        database name = %B-h.e
        database type = exodusII
        At time {t_0_0}, increment = 0.001
        global time as mom_bal_time
        global wall_clock_time as run_time
        global mat_time as time
        global vp_rate_scale_factor
        global roof_disp
        global floor_disp
        global right_wall_disp
        global contact_energy
        global external_energy
        global internal_energy
        global kinetic_energy
      end history output output_3

      #--------- Restart ---------

        begin restart data restart_1
            database name = %B.rsout
            overwrite = True
            at wall time 0.0h increment = 12.0h
        end restart data restart_1

      #--------- Solver ---------

      Begin solver

      end solver

  end presto region SmRegion.1

end adagio procedure The_Procedure.1

end sierra WIPP Isothermal Disposal Room
```

D.3.2. Container Compaction Simulations

```
begin sierra WIPP Isothermal Disposal Room

   title Simulation of WIPP Isothermal Disposal Room with CCO disposal canisters

   restart = automatic

   # ============================ 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"
   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


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

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

begin solid section solid_cco
   formulation = selective_deviatoric
   strain incrementation = strongly_objective
end

begin solid section solid_cco_wood
   formulation = selective_deviatoric
   strain incrementation = strongly_objective
end

begin shell section shell_t7p112mm #
   thickness = 0.007112 # 0.280 in.
   lofting factor = 0.5
end

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-1_cco_row_2017_11_28_nocut_thickerRoom_new.g  # added additional
   ↪  nodesets for getting average room closure
   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 cco_drum_wall   cco_drum_base   cco_drum_lid \#
      cco_drum_wall_2 cco_drum_base_2 cco_drum_lid_2
      material ASTM_A1008_Drum_Steel
      solid mechanics use model ml_ep_fail
      section = shell_t1p5mm
   end

   begin parameters for block cco_cv_wall cco_cv_wall_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      #solid mechanics use model bcj_mem
      section = shell_t7p112mm
   end

   begin parameters for block cco_cv_base   cco_cv_top_flange   cco_cv_lid \#
      cco_cv_base_2 cco_cv_top_flange_2 cco_cv_lid_2
      material SA-240_304L
      solid mechanics use model ml_ep_fail
      #solid mechanics use model bcj_mem
      section = solid_cco
   end

   begin parameters for block cco_wood_lower   cco_wood_upper \#
      cco_wood_lower_2 cco_wood_upper_2
      material Plywood
      solid mechanics use model elastic_fracture
      #solid mechanics use model elastic
      section = solid_cco_wood
   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 = cco_cv_top_flange   cco_cv_lid   cco_cv_wall   cco_cv_base \#
         cco_cv_top_flange_2 cco_cv_lid_2 cco_cv_wall_2 cco_cv_base_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 cco_surf
      block = cco_drum_wall  cco_drum_base cco_drum_lid \#
         cco_wood_lower cco_wood_upper \#
      cco_cv_wall    cco_cv_base   cco_cv_top_flange   cco_cv_lid
   end

   begin contact surface cco_surf_2
      block = cco_drum_wall_2  cco_drum_base_2 cco_drum_lid_2 \#
         cco_wood_lower_2 cco_wood_upper_2 \#
      cco_cv_wall_2    cco_cv_base_2   cco_cv_top_flange_2  cco_cv_lid_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

#In February 2020, we realized this command contains a mistake.  It was
#intended to only specify mu_s = 0.5 between the room surface and each clay
#seam surface, but it also set mu_c = 0.5 for clay seam F and G.
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 = cco_surf
   friction model = metal_to_salt_friction_model
end

begin interaction metal_to_salt_2
   master = room_surf
   slave = cco_surf_2
   friction model = metal_to_salt_friction_model
end

begin interaction metal_to_metal
```

```
      # self contact
      surfaces = cco_surf cco_surf
      friction model = metal_to_metal_friction_model
   end

   begin interaction metal_to_metal_2
      # self contact
      surfaces = cco_surf_2 cco_surf_2
      friction model = metal_to_metal_friction_model
   end
   begin interaction metal_to_metal_3
      # contact between 2 groups of canister blocks
      surfaces = cco_surf cco_surf_2
      friction model = metal_to_metal_friction_model
   end
   begin interaction metal_to_analytic_plane
      # contact between full drum blocks and plane
      surfaces = cco_surf cco_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 cco_wood_failure
   block = cco_wood_lower   cco_wood_upper \#
   cco_wood_lower_2 cco_wood_upper_2
   material criterion = on kill when 1 integration points remain
end

begin element death cco_drum_failure
   block = cco_drum_wall   cco_drum_base   cco_drum_lid \#
   cco_drum_wall_2 cco_drum_base_2 cco_drum_lid_2
   material criterion = on kill when 1 integration points remain
end

begin element death cco_cv_failure
   block = cco_cv_wall   cco_cv_base   cco_cv_top_flange   cco_cv_lid \#
   cco_cv_wall_2 cco_cv_base_2 cco_cv_top_flange_2 cco_cv_lid_2
   material criterion = on kill when 1 integration points remain
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

## ~~~~~~~~~~~~~~~~~~ Results Output ~~~~~~~~~~~~~~~~~~ ##

begin results output output_1
   database name = efiles/%B-r.e
   database type = exodusII
   #Output data at 0, 10, 100, and 1000 years of material time
   $-10
   $0
   additional times = 0
   $10
   additional times = 0.5617615152
   $100
   additional times = 5.589456503
   $1000
   additional times = 5.617615152
   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


         ## ~~~~~~~~~~~~~~~~~~~ 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
```

DISTRIBUTION

Email—External

Name	Company Email Address	Company Name
Virginia Kay	virginia.kay@nnsa.srs.gov	National Nuclear Security Administration
Bret Brickner	bricknerbd@ornl.gov	Oakridge National Laboratory
Maxcine Maxted	maxcine.maxted@srs.gov	Savannah River Site
Joseph D'Amelio	joseph.damelio@srs.gov	Savannah River Site
Ron Livingston	ron.livingston@srs.gov	Savannah River Site
Brad Day	Brad.Day@wipp.ws	Nuclear Waste Partnership

Email—Internal

Name	Org.	Sandia Email Address
T. Shelton	1542	trshelt@sandia.gov
J. Blecke	1550	jblecke@sandia.gov
J. Bean	1554	jebean@sandia.gov
A. Brundage	1554	albrund@sandia.gov
J. Rath	1555	jsrath@sandia.gov
J. Smith	1556	jasmith@sandia.gov
J. Pott	1557	jpott@sandia.gov
S. Nelson	1558	smnelso@sandia.gov
B. Salzbrenner	7617	bsalzbr@sandia.gov
E. Matteo	8842	enmatte@sandia.gov
K. Kuhlman	8842	klkuhlm@sandia.gov
R.C. Camphouse	8842	rccamph@sandia.gov
M. Mills	8842	mmmills@sandia.gov
M. Starr	8853	mjstarr@sandia.gov
D. Conley	8912	dconley@sandia.gov
B. Park	8912	bypark@sandia.gov
S. Sobolik	8912	srsobol@sandia.gov

Name	Org.	Sandia Email Address
T. Zeitler	8912	tzeitle@sandia.gov
A. Holland	8914	aaholla@sandia.gov
S. Broome	8914	stbroom@sandia.gov
C. Herrick	8914	cgherri@sandia.gov
S. Bauer	8916	sjbauer@sandia.gov
P. Shoemaker	8880	peshoem@sandia.gov
D. Kicker	8880	dckicke@sandia.gov
G. Duran	8880	gasosa@sandia.gov
R. Kirkes	8881	grkirke@sandia.gov
S. Wagner	8881	swagner@sandia.gov
Technical Library	1911	sanddocs@sandia.gov



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.