VISCOELASTIC EFFECTS ON A POLYETHERIMIDE CYLINDER WITH CONSTANT RADIAL DEFORMATION

by

L. EDWARD PARKMAN III

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

Acknowledgements

First, my wife: Katie is quite possibly the most patient person I know. She's sacrificed uncountable amounts of her own free time while I was holed up studying or working on that one last important assignment. I love you and I thank you.

I'd also like to dedicate this to my kids: Quade, Ruby, Henry, and Eleanor. You guys are wonderful and you make me smile on a daily basis. I love you.

Many thanks go to my advisor, Dr. Ashfaq Adnan. Thank you for offering the Advanced Mechanics of Materials class and for all of your help with this work. I'd also like to thank Dr. Wen Chan and Dr. Kent Lawrence from my committee for their efforts on my behalf.

I also need to thank Corning Optical Communications LLC. I've been employed at Corning for a little more than 10 years. They also funded my graduate studies and allowed the schedule flexibility to make classes.

Specifically:

Eric Carter, Jeff Ray, Dr. Martin Schulte, and Jeff Ferner – thank you for your support, confidence, and advice over the years.

John Vansickle, Mike Fricker, Mike Juneau, Roger Peterson – thank you for making and testing my samples. Thank you more for your great (and continual) advice on how to get it done well.

Dr. Andrey Kobyakov and Dr. Esteban Marin – thank you for your technical guidance with study subjects and so much help with good background literature.

Mike de Jong – thank you for more than a decade of mentorship and friendship. I appreciate the extra work you've put in to help me out with this and so many other things.

April 6, 2015

Abstract

VISCOELASTIC EFFECTS ON A,
POLYETHERIMIDE CYLINDER
WITH CONSTANT RADIAL
DEFORMATION

L. Edward Parkman III, MSME

The University of Texas at Arlington, 2015

Supervising Professor: Ashfaq Adnan

The relaxation of an axisymmetric radially crimped joint is analyzed with the inner component being formed from the engineering grade polymer ULTEM™ 1000 polyetherimide. Creep and Relaxation tests were performed at several temperatures in order to use Time Temperature Superposition theory to assemble a master curve for predictions at long times based on short time testing.

The master curves for each test were fit to the appropriate Prony Series and all coefficients are tabulated for re-use. The elastic theory for the classic thick walled cylinder was reviewed and the conversion from elasticity to viscoelasticity was demonstrated. Strength predictions are shown and compared to their elastic counterparts. Finally, a non-dimensionalized model of strength over time is presented.

Table of Contents

List of Illustrations

List of Tables

List of Equations

Chapter 1

Problem Statement

1.1 Geometry

Different types of crimped joints have been used in several industries for many years. Examples include metal caps on glass beverage bottles, lids for canned foods, and cartridge attachment for ammunition. Radial crimping of tubular members is very common for attaching fittings for various hydraulic & pneumatic applications as well as attaching connectors to cables in the Telecommunications industry.

Figure 1-1 below shows a schematic view of a radially crimped joint. The yellow metal cylinder is originally slip fit over the blue plastic cylinder. A tool is used deform the outer cylinder radially inward until a press fit is developed between the cylinders. For the press fit to remain after the crimp tool is removed, the outer ring material must be deformed past its yield point. As the crimp tool is removed, the outer ring will expand radially outward as the elastic stress in the part is relieved, leaving a press fit that is related to the plastic strain induced in the outer crimp ring.

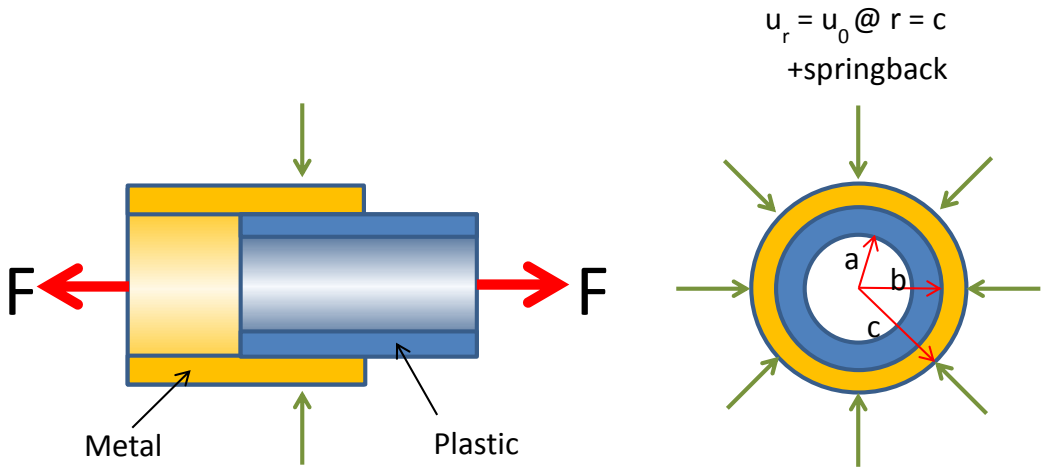


Figure 1-1: Schematic View of Crimped Joint (x-section on left)

Crimping was originally used to join thin metal parts, but it has also been used to join dissimilar materials, such as shown in **Figure 1-1**. This work is focused on examining and developing a predictive model for the long term effects of choosing a plastic material for the inner cylinder in a crimped joint. In this application, the prediction of the long-term relaxation of the retention force F is the goal.

For the sake of this work, the initial retention force F_i is assumed to be known shortly after the crimping operation (t≈0), and the specification for retention force F can be expressed as a percentage of F_i. Assuming an original design safety factor of 1.5 gives a specification for F_{min} = 66% of F_i. Radius a is assumed to be 1.0 mm and b is 1.5 mm. Radial deformation u_r is assumed to be 0.075 mm, corresponding to 5% strain in the radial direction.

For this work, the planned prediction time will be 20 years at 23˚C. If possible, prediction to 20 years at 40˚C would be desirable to show the effects of temperature and highlight the difference between products designed for indoor and outdoor applications.

1.2 Assumptions

In developing a material model, it's good practice to start simple, check against reality, and then increase complexity. In this work, uniaxial properties are measured and used to predict the effects of time on a simplified version of a crimp joint.

1.2.1 No Local Deformations

For this work, it will be assumed that the entire length of the outer ring is radially deformed inwardly. End effects of the crimp are also not covered. By making this assumption, the radial stress at the interface between the rings is constant and a 2D approximation can be used.

This assumption ignores local deformations such as radial dimples and end effects that act as a stress concentrator. For this reason, the methodology discussed herein is focused on predicting relaxation and will not be a good tool for predicting local failure modes such cracking at the ends due to localized stress.

1.2.2 Elastic Deformation Only

This model is intended to predict linear viscoelastic effects only. This methodology should only be applied to stress/strain conditions below yield. One exception would be if a linear elastic perfectly plastic (LEPP) material model fits the material choice for the inner cylinder. If an LEPP model fits, then similar methods may be useful for predicting the viscoelastic relaxation from yield. Because many plastic materials have non-linear stress-strain behavior, this work was limited to viscoelasticity. Any initial material yielding would very likely cause this model to predict artificially high retention strengths over time (dependent on material yield behavior).

1.2.3 $E_{Metal} >> E_{Plastic}$

By assuming that the stiffness of the metal outer ring is much greater than the stiffness of the inner plastic material, a constant radial deformation can be assumed because the amount of extra deformation for every unit of relaxation is relatively small. This methodology should not be used for applications where the total stiffness of the outer ring is similar to the total stiffness of the inner ring (*including* the thickness of each). If the two rings had similar stiffness, then as the inner ring relaxed, the outer ring would deform it more (non-constant radial deformation).

1.2.4 Retention Force is a Function of Interface Pressure Only

Expressed as an equation,

Equation 1-1: Pull Force for Radially Deformed Cylinder

$$F = \mu_s A \sigma_r(r = b)$$

In Equation 1-1, F is the retention force as shown in **Figure 1-1**. The static friction coefficient is represented by μ_s and A is the circumferential contact area. The pressure at the interface is identical to the radial stress at the outside diameter boundary, which is represented by σ_r.

1.3 Simplified Problem

Using the assumptions discussed in section 1.2 above, the problem at hand can be reduced to a cylinder under constant deformation, as seen in **Figure 1-2**. In **Figure 1-2**, u_r is the strain in the radial direction and is considered to be constant at the outside diameter boundary.

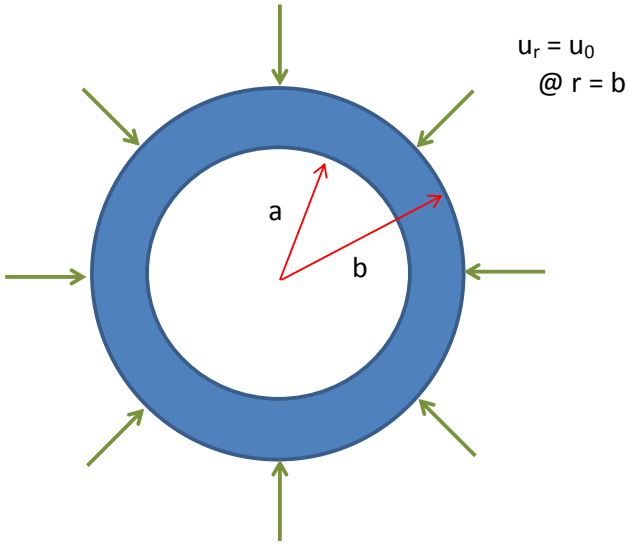


Figure 1-2: Problem Simplified to a Cylinder with Constant Radial Deformation

1.4 Materials

Polyetherimide (PEI) is a widely used amorphous engineering grade thermoplastic with high strength, high stiffness, high service temperature, inherent flammability resistance, and good chemical resistance. ULTEM™ 1000 was chosen for this study because it is PEI with no modifiers, fillers, or reinforcement. According to SABIC's website [1], ULTEM™ 1000 is used in the Aerospace, Transportation,

Healthcare, and Wire & Cable industries. According to the datasheet on the same website, the T_g of ULTEM™ 1000 is 217˚C.

1.4.1 Stress/Strain Behavior

In **Figure 1-3**, both the engineering and calculated true stress-strain curves are plotted. The engineering stress-strain data plotted in **Figure 1-3** is courtesy of SABIC's website [2] and the true stress-strain was calculated. The equations for true stress and strain are shown in **Equation 1-2** (Ugural and Fenster [3]). True stress and strain are denoted by σ and ε, while the engineering stress and strain are σ_0 and ε_0.

Equation 1-2: Equations for True Stress and Strain [3]

$$\varepsilon = \ln(1 + \varepsilon_0)$$

$$\sigma = \sigma_0(1 + \varepsilon_0)$$

ULTEM™ shows ductile stress-strain characteristics common in thermoplastics. Bilinear approximations or assumptions of LEPP material models will have errors in the neighborhood of yield, but may be acceptable for some applications, such as those that have strain/displacement as an input or limit. The 5% strain corresponding to the constant displacement discussed in the problem definition (section 1.1) shows to be in the elastic region, meeting the assumptions in section 1.2.2 (linear viscoelastic).

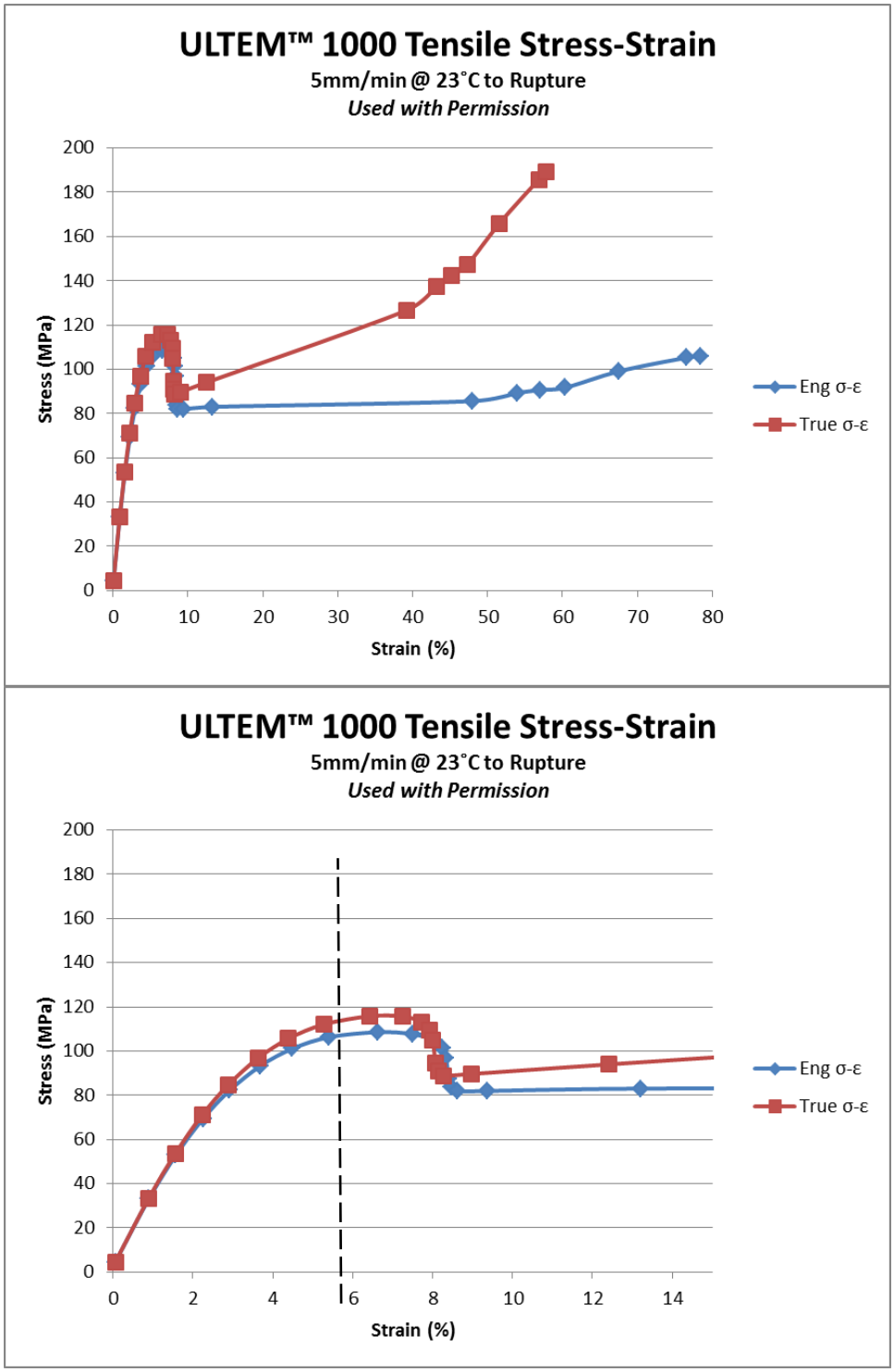


Figure 1-3: Stress-Strain Based on Data from SABIC's Website [2]

1.4.2 Tensile Creep Behavior

Figure 1-4 shows a plot created with Tensile Creep data from SABIC's website [2]. At the onset of this work, the number of datasets at equal stress was a concern, and it was unknown how far into the future this data would enable prediction. Upon later re-evaluation, the 14MPa set from SABIC overlaps nicely and would definitely predict farther than 20 years into the future at 23˚C. It was also *believed* that Relaxation data would be more valuable due to its *perceived* ease in analytical solutions (see section 4.2 for more details). In section 3.4, Creep results from this work are compared to SABIC's data.

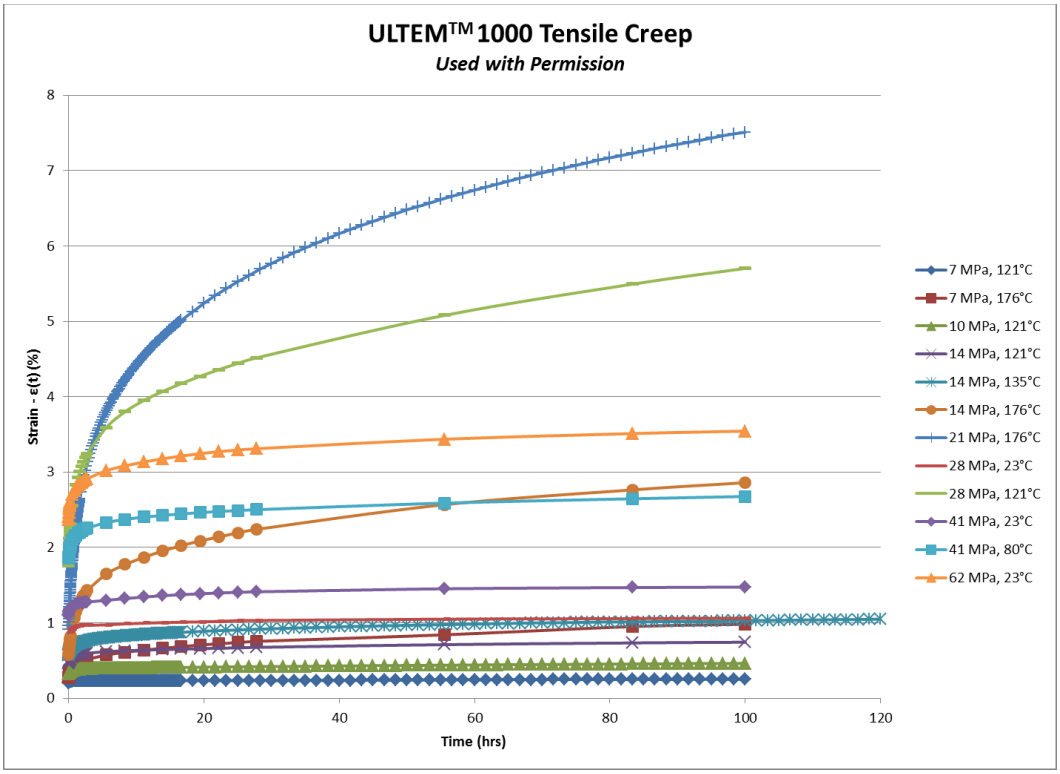


Figure 1-4: Tensile Creep Based on Data from SABIC's Website [2]

1.4.3 Dynamic Mechanical Analysis

Figure 1-5 shows a plot created with data from SABIC's website [2]. Dynamic Mechanical Analysis inputs a cyclical load on a test sample and measures the output cycles amplitude and frequency lag form the original input. E' is the storage modulus, E'' is the loss modulus, and tan delta is the ratio E'/E''. The complex modulus $E^*(\omega) = E'(\omega) + iE''(\omega)$. With this data, the viscoelastic behavior can be related in the Fourier domain (see [4] for more info).

This data illustrates one method for determining T_g. For reference 217°C ~ 422°F. A local peak in E'' can be seen at T_g. The original intention was to use this data for predictions, but scans of *several frequencies* over different temperatures are needed to predict long time effects. This data was all taken at 6.28 Hz.

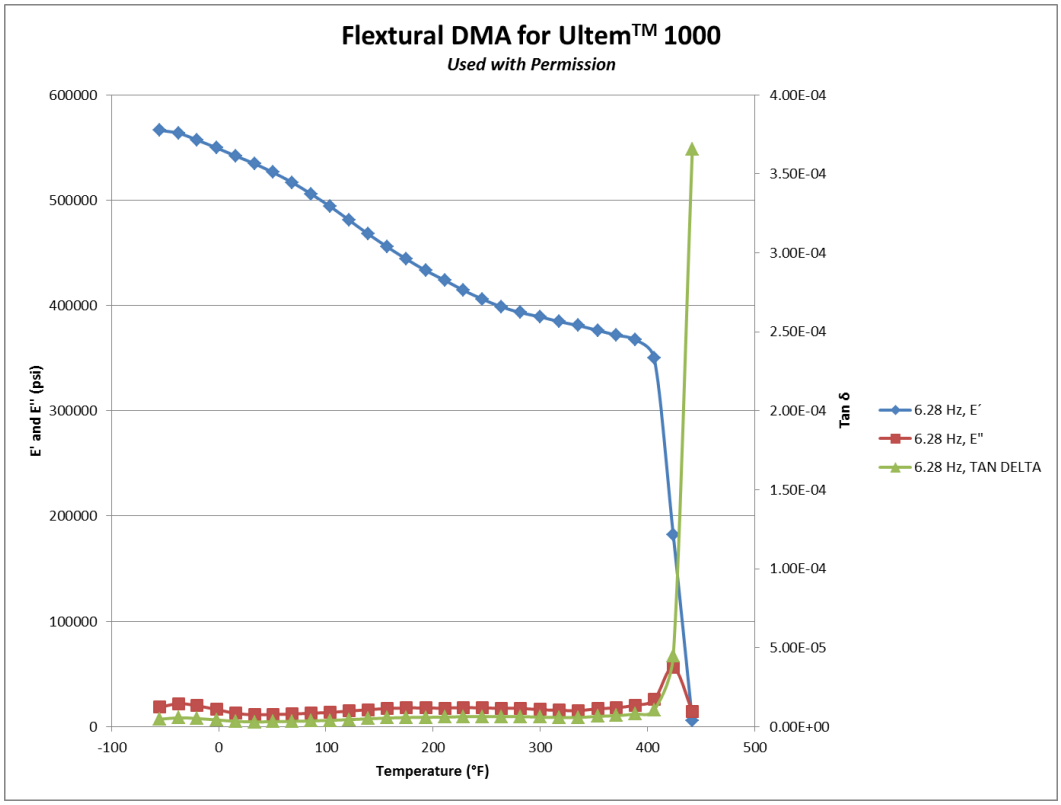


Figure 1-5: Flexural DMA Based on Data from SABIC's Website [2]

Chapter 2

Background Information

Because the focus of this work was related to time effects, empirical data for the structural properties of PEI was needed to predict long-term effects. As collecting information over several years is impractical, Time-Temperature-Superposition-Principle (TTSP) methodology was chosen in order to collect short term data and make long term predictions. As discussed by Brinson and Brinson [4], TTSP was proposed by H. Leaderman in the early 1940's and the method has been refined by several people over the years. References to these works can be found in their book [4].

The Relaxation test was chosen due to the Relaxation Modulus E(t)'s similarity to Young's Modulus E and its simplicity with analytical solutions. The Creep test was additionally chosen because of a desire to compare with SABIC's data and because of a misunderstanding of inputting D(t) data into commercially available Finite Element Analysis (FEA) software (D(t) vs D*(t)). **Figure 2-1** shows a comparison of the inputs and outputs for each of these tests. **Equation 2-1** and **Equation 2-2** show the meanings of E(t), D(t), and their relations. **Equation 2-2** indicates that E(t) and D(t) are not inverses in the time domain, but are inverses in the Laplace domain. Further explanation is supplied by Brinson [4] and Christenson [5].

In these equations, E(t) is the Relaxation Modulus and D(t) is the Creep Compliance. $\sigma(t)$ is the measured time dependent stress from an input constant strain ε_0 and $\varepsilon(t)$ is the time dependent strain from an input constant stress σ_0. E(s) and D(s) (with bars above) are the Laplace transforms of E(t) and D(t).

Equation 2-1: Relaxation Modulus E(t) and Creep Compliance D(t)

$$E(t) = \frac{\sigma(t)}{\varepsilon_0}; \quad D(t) = \frac{\varepsilon(t)}{\sigma_0}$$

Equation 2-2: Relation between Relaxation Modulus and Creep Compliance

$$E(t) \neq \frac{1}{D(t)}; \quad \bar{E}(s) = \frac{1}{\bar{D}(s)}$$

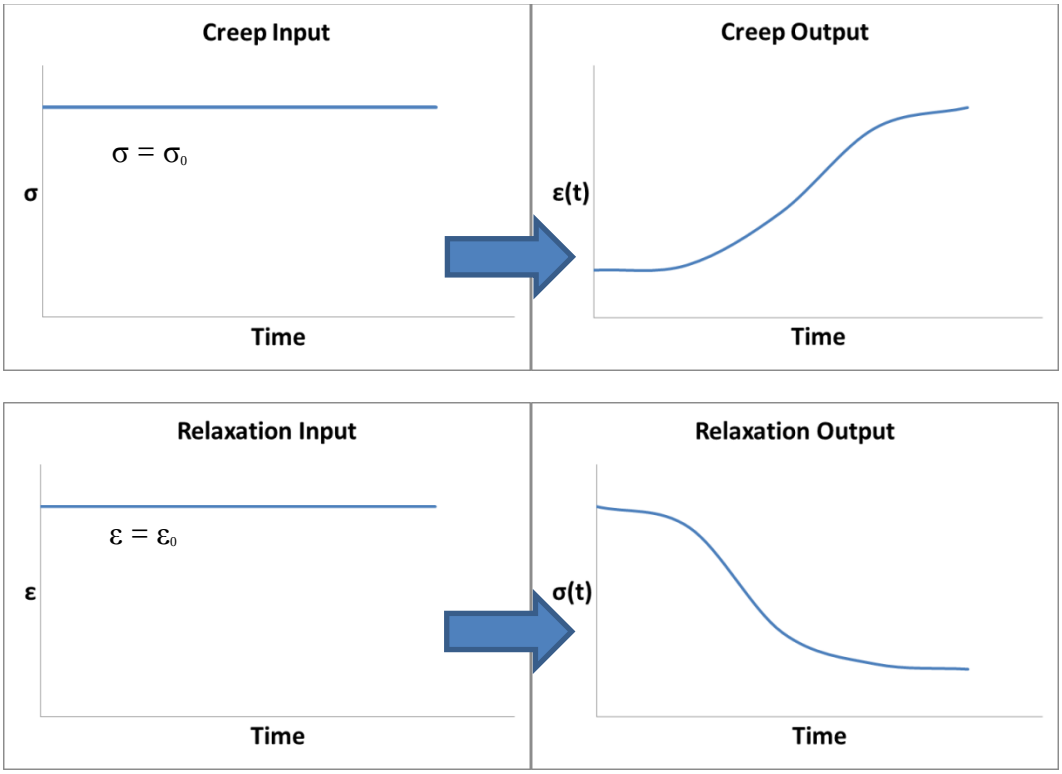


Figure 2-1: Illustrated Comparison of Creep and Relaxation Tests

Figure 2-1 shows the difference between the Relaxation and Creep tests. For the Creep test, a constant stress is applied and the strain is measured over time, similar to hanging a weight from a test specimen. For the Relaxation test, a constant strain is supplied and the stress is measured over time. Note that the illustrations in **Figure 2-1** assume a viscoelastic solid model (as opposed to a V.E. liquid). The long term behaviors tend towards 0 or ∞ if a V.E. liquid. Testing (discussed later), indicates that a much more complicated mathematical model with multiple parameters is needed to characterize ULTEM™ PEI.

From empirical data, a mathematical function is fit to the time dependent material parameter (different functions for Creep or Relaxation, discussed later). Once expressed as a function, the Laplace transform can be taken. If the elastic solution for the problem is known, the Laplace transforms of the time dependent stress and strain are related by that function, as seen in **Equation 2-3** and **Equation 2-4**. In simple terms, the stress-over-time answer to this problem is the inverse Laplace transform of the strain times s (Laplace domain variable) times the Laplace transform of the Relaxation Modulus.

Equation 2-3: Relation of Stress and Strain in Viscoelasticity

$$\bar{\sigma} = \bar{E}^{*}\varepsilon\,; \quad \bar{\varepsilon} = \bar{D}^{*}\sigma$$

Equation 2-4: Definition of Complex Moduli

$$\bar{E}^{*} = s\bar{E}\,; \quad \bar{D}^{*} = s\bar{D}$$

Chapter 3

Testing

3.1 Test Plan

In planning to use TTSP, some information was needed to relate time and temperature. Williams-Landel-Ferry (WLF) and Activation Energy methodologies were investigated. According to Brinson [4], the slope of the curves generated by the acceleration factors (a_T) are discontinuous at T_g. The WLF equation works better above T_g and the Activation Energy equation (**Equation 3-1**) works better below T_g. It was observed that if the Arrhenius Activation Energy were known at some T_{ref}, then the acceleration factor for any temperature could be calculated using **Equation 3-1**. It's worth noting that the E_a used below is specific to this failure mode. Other E_a values, such as the E_a for thermal decomposition, do not apply to this mode. **Equation 3-1**, R is the universal gas constant.

Equation 3-1: Acceleration Factors for TTSP (Arrhenius Model) [4]

$$\ln a_T = -\frac{E_a}{R}\left(\frac{1}{T} - \frac{1}{T_{ref}}\right)$$

Foreman et al. [6] investigated creep in PEI and found the best fit Arrhenius Activation Energy to be approximately 92 kJ/mol at 67˚C. As shown in section 3.4, the testing for this work resulted in similar E_a *for Creep*. Using this information, the first test plan (see **Table 3-1**) was set by simple trial and error using a simple spreadsheet. The important parameters for the test are that the tests are short, all curves overlap each other, and all tests must be conducted at less than T_g for **Equation 3-1** to be applicable.

The test plan shown in Table 3-1 would have allowed for prediction of behavior in excess of 20 years at 40˚C.

Table 3-1: Original Test Plan

								Time-Temp Superposition			Time-Temp Superposition		
Temp (°C)	Start Time (s)	End Time (min)	End Time (sec)	Temp (K)	a_T (--> 67°C)	a_T (23°C --> 67°C)	a_T (40°C --> 67°C)	Start Time @ 23°C (s)	End Time @ 23°C (s)	End Time @ 23°C (yrs)	Start Time @ 40°C (s)	End Time @ 40°C (s)	End Time @ 40°C (yrs)
200	1	80	4800	473.2	9251.82	0.008	0.0607	1.16E+06	5.54E+09	175.68	1.52E+05	7.31E+08	23.17
160	1	10	600	433.2	1070.26	0.008	0.0607	1.34E+05	8.02E+07	2.54	1.76E+04	1.06E+07	0.34
120	1	10	600	393.2	79.8251	0.008	0.0607	9.97E+03	5.98E+06	0.19	1.31E+03	7.89E+05	0.02
80	1	10	600	353.2	3.3068	0.008	0.0607	4.13E+02	2.48E+05	0.01	5.44E+01	3.27E+04	0.00
23	1	15	900	296.2	0.00801	0.008	0.0607	1.00E+00	9.00E+02	0.00	1.32E-01	1.19E+02	0.00

During testing, it was discovered that the equipment was not capable of reliably holding temperatures above 113˚C, which is discussed further in section 3.3. For this reason, the test plan needed to be quickly modified. The original spreadsheet was re-used to re-plan the test with the new max temperature of 110˚C. The resulting test plan can be seen in **Table 3-2**. Unfortunately, predictions at 40C are not supported by the revised test plan because Creep predictions of 20 years at 40˚C would require testing for approximately 2 weeks at 110˚C (delivery of work would not be possible by the due date). Note that an intermediate test at 40˚C had to be added because there was no overlap in the 23˚C and 80˚C test results.

Table 3-2: Revised Test Plan

								Time-Temp Superposition			Time-Temp Superposition		
Temp (°C)	Start Time (s)	End Time (min)	End Time (sec)	Temp (K)	a_T (--> 67°C)	a_T (23°C --> 67°C)	a_T (40°C --> 67°C)	Start Time @ 23°C (s)	End Time @ 23°C (s)	End Time @ 23°C (yrs)	Start Time @ 40°C (s)	End Time @ 40°C (s)	End Time @ 40°C (yrs)
110	1	2880	172800	383.2	38.3286	0.008	0.0607	4.79E+03	8.27E+08	26.20	6.31E+02	1.09E+08	3.46
80	1	10	600	353.2	3.3068	0.008	0.0607	4.13E+02	2.48E+05	0.01	5.44E+01	3.27E+04	0.00
40	1	10	600	313.2	0.06073	0.008	0.0607	7.58E+00	4.55E+03	0.00	1.00E+00	6.00E+02	0.00
23	1	15	900	296.2	0.00801	0.008	0.0607	1.00E+00	9.00E+02	0.00	1.32E-01	1.19E+02	0.00

By observing the differences in the two test plans, the power of TTSP is evident. In converting to 23˚C, 12 minutes at 200˚C gives the same future prediction as 48 hours of testing at 110˚C for PEI.

3.2 Test Samples

Samples were injection molded from ULTEM™ 1000 into a standard 1 mm thick impact disk. Samples were then machined on a Computer Numerical Controlled Mill to the design shown in **Figure 3-1** and Appendix A. The sample were molded and machined by Corning Optical Communications LLC (Corning) personnel using Corning equipment with permission.

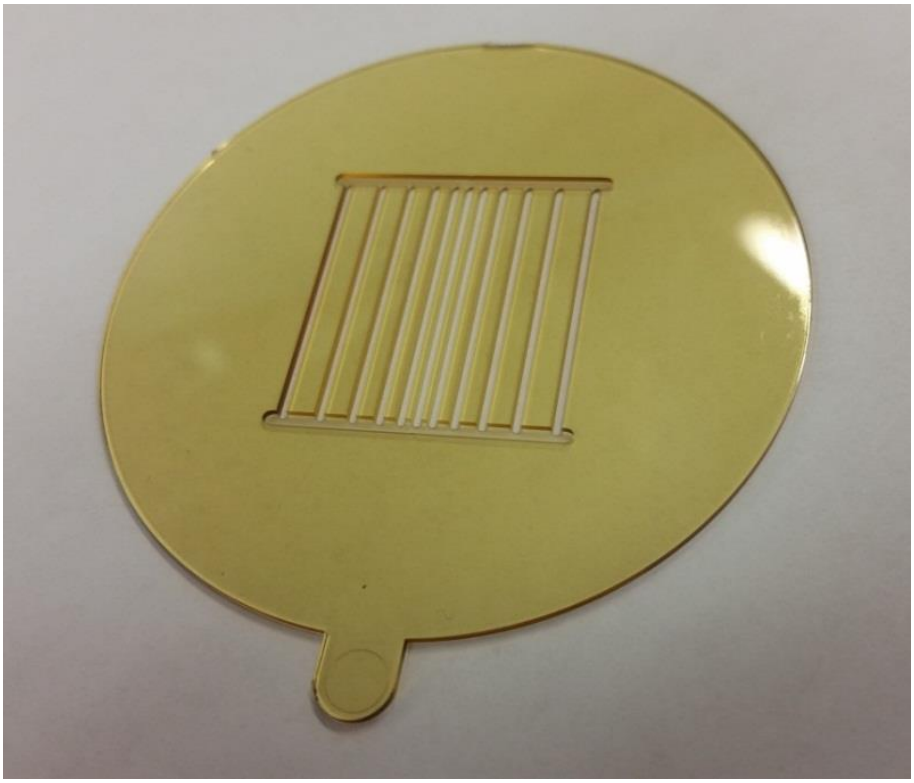


Figure 3-1: Test Samples

The molded and machined test samples can be seen in **Figure 3-1**. This design was optimized to allow it to be cut with a standard 1/32" diameter endmill. At the time that these samples were designed, the amount of vibration in the cutting process was unknown, so samples were cut at various widths with the intention of utilizing the smallest size with good cut quality.

Figure 3-2 shows a magnified view of one of the smallest (0.5 mm) sample's cut surface, which was compared and found to be similar quality to all other sizes. **Figure 3-2** was taken using a stereo microscope, and the smallest size was used for testing. The test samples were left attached to the original impact disk, and this was found to be a very effective way to protect the samples during shipping. Optical measurement of the samples indicated a parallelism of approximately 0.020 mm.

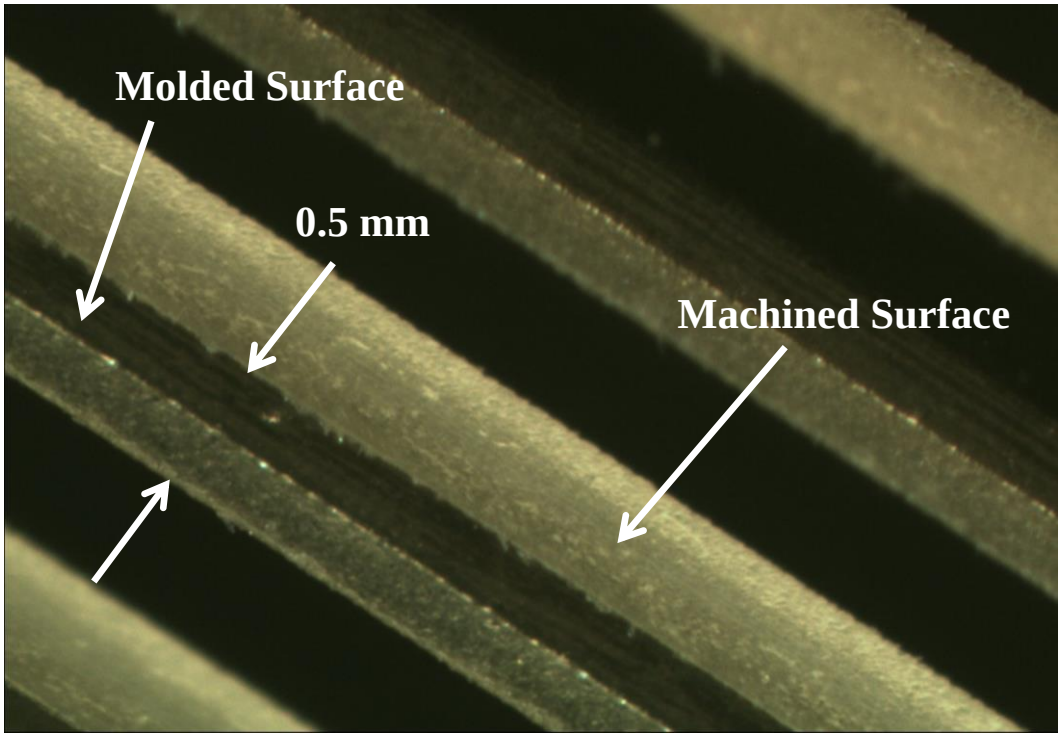


Figure 3-2: Magnified View of Cut Surface on 0.5mm Sample

3.3 Equipment

The equipment used for this testing was a TA Instruments brand Dynamic Mechanical Analyzer (DMA) model Q800. Full details on this model and all of the machine specifications in this section are directly from TA Instruments' website [7]. The machine was set to apply static tension or strain for a standard Creep or Relaxation test (not a dynamic measurement). The DMA used is property of Corning, was operated by Corning personnel, and was used with permission.

The major machine specifications that drove testing parameters were the advertised load cell rating of 18 N and the temperature rating of -150 to 600˚C. The Young's Modulus from SABIC's literature [1] was used to ensure that the sample geometry combined with the input strains for the Relaxation test would not overload the load cell. For the Creep test, the equivalent initial stress was calculated using the same Young's Modulus and the design nominal geometry.

Originally, the test plan included samples tested at 200˚C and shorter times. For some reason, this machine would not reliably hold temperatures above 113˚C. Unfortunately, adequate time to troubleshoot was not available and the test plan was modified to use a max temperature of 110˚C for a longer time. In later troubleshooting, it was observed that this machine is commonly used for materials with a low T_g and has been upgraded with an aftermarket intercooler which limits the max temperature (not a concern for the typical applications for this specific unit).

3.4 Results

3.4.1 Creep

Figure 3-3 shows the results of measured Creep Compliance D(t) at four temperatures. When evaluating the applicability of TTSP for Creep or Relaxation data, one major need is for the curves to overlap when the data is moved through time. Note that changes in density were not considered for this work adding a small amount of error, but could be added in future works.

As can be seen graphically, the only overlap between the 23˚C and 80˚C curves is between the first and last datapoints. This is problematic for two reasons. The first is that TTSP should generate a smooth curve through time. If only single datapoints are used, a discontinuity could exist that would not be obvious.

The second is related to this specific data. The first datapoint for each temperature curve has a distinctive difference in curvature, including the appearance of an unanticipated inflection point early in each curve. Because this datapoint is very quickly collected after the test started and because realistic testing is unable to produce instantaneous results as mathematically modeled by the Heaviside function, it is strongly believed that the machine and sample have not achieved/steadied to the desired stress. For this reason, the first datapoints will be neglected in later dealings with this data.

If the response at shorter times is needed for a different application, attempting to push the equipment to measure at a faster measurement frequency is not recommended. Instead, running a sample set at a temperature less than T_{ref} will cause that dataset to have an acceleration factor a_T that is less than 1. In other words, a data point collected at $T<T_{ref}$ one minute into the test may tell you how that material would respond after one second at T_{ref} (for illustration only).

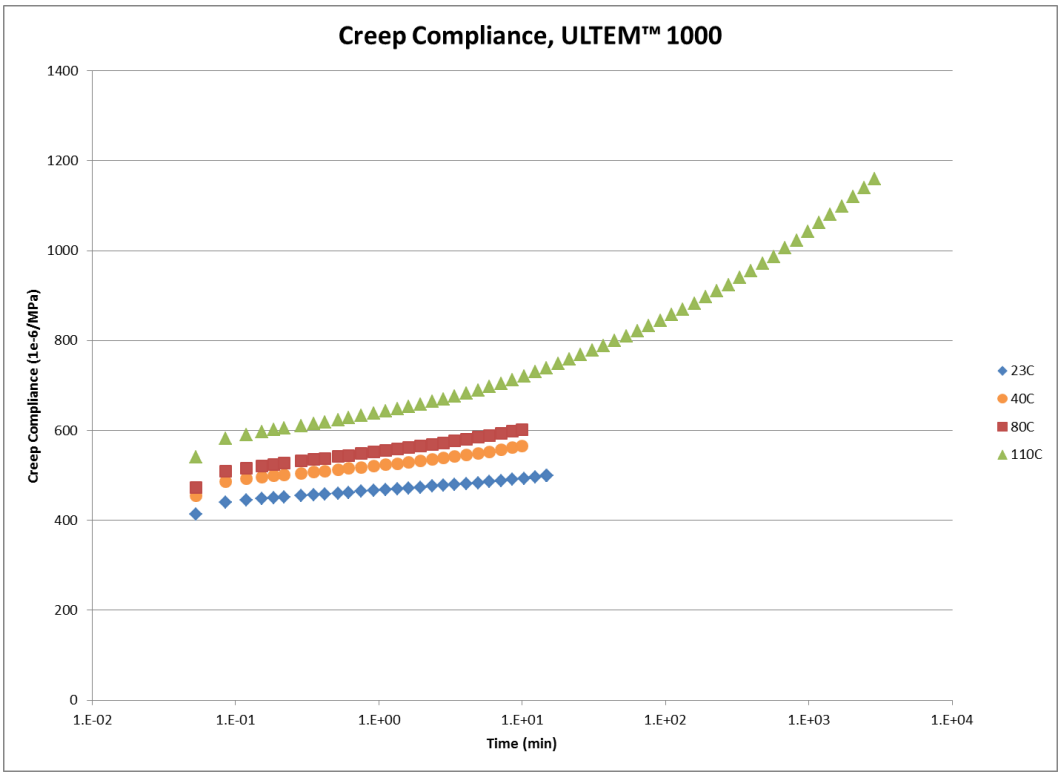


Figure 3-3: Creep Compliance Results

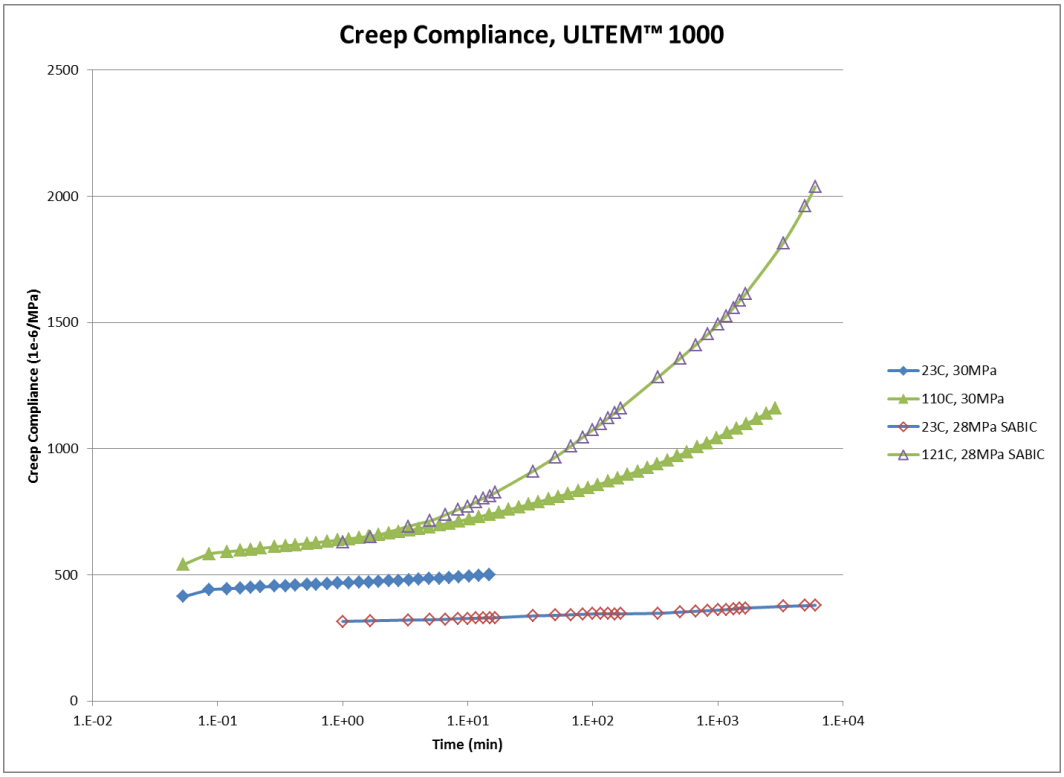


Figure 3-4: Creep Compliance Results Compared with SABIC's Data [2]

For verification purposes, the test data was compared to SABIC's Creep data as seen in **Figure 3-4**. Unfortunately, these tests were run at different conditions. When comparing the sets of 23˚C data shown with blue lines, it is to be expected that the observed time dependence would be "slower" with lower stress. SABIC's data shows lower compliance (higher effective stiffness) with lower stress applied. Without more data at lower temperatures or shorter times, it's not possible to draw a solid conclusion, but this relation is at least directionally correct.

The green curves show the higher temperature test results. SABIC's test was conducted at slightly lower stress and 11˚C higher temperature than the test from this work. The expected relation for this situation would be for the SABIC data to show a

lower compliance at very short times due to the lower stress and a faster acceleration of compliance at longer times due to the higher temperature. In fact, that is exactly what is shown in **Figure 3-4**. This new test data can be said to be very similar to the original data from SABIC.

3.4.2 Relaxation

The Relaxation test also showed some unexpected behavior at very short times. As seen in **Figure 3-5**, unexpected curvature can be seen in the first few datapoints of each set. This is believed to be the same settling phenomenon observed in section 3.4.1. Note that the additional 40˚C test was requested after observing a gap in information between the 23˚C and 80˚C curves. The 40˚C test was planned using the activation energy method from the tests by Foreman et al. [6] with the intention of filling this gap. In the case of Relaxation, this 40˚C test did not fill the gap and supplies less long-term information than the 23˚C test.

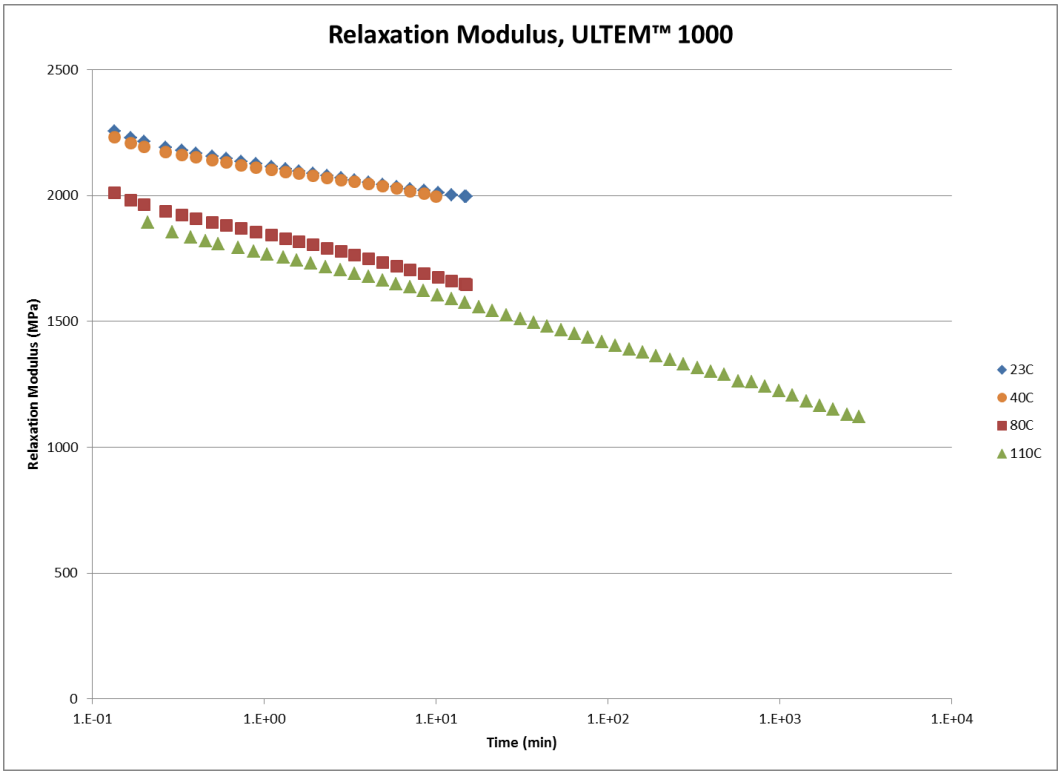


Figure 3-5: Relaxation Modulus Results

3.5 Time-Temperature Superposition

A fast and repeatable method was desired for aligning the curves in time. To accomplish this, a simple program was written to linearly interpolate the shifted time of the higher temperature compliance between the two nearest compliances of the lower temperature curve. Once the first shift factor a_T is found, the method can be repeated for subsequent higher temperature tests to assemble the master curve. This program and its results can be found in Appendix B. The error associated with a linear interpolation instead of an exponential curve fit is limited by using the two nearest compliances. Note that because of the apparent equipment stabilization behavior in the first datapoints, the

fourth point was chosen due to its distance from the odd behavior and available overlap between curves.

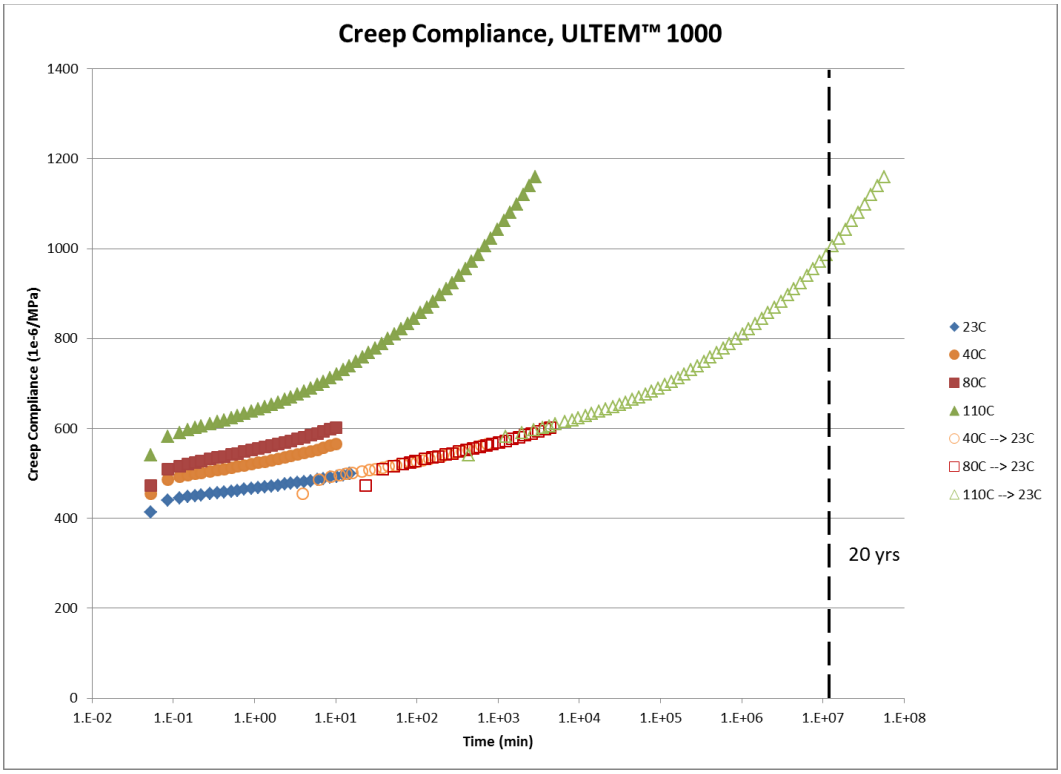


Figure 3-6: Creep Compliance Results and TTSP Master Curve

As seen in **Figure 3-6**, the assembled Creep Master Curve allows predictions in excess of 20 years. The acceleration factors and calculated activation energy E_a are shown in **Table 3-3**. It's important to note that the difference in the original acceleration factors formulated by lining up curves visually on an automated graph and those generated by linear interpolation varied by *3-14%*, which may have a very large difference at long times. The calculated activation energies (using **Equation 3-1**) for

80˚C and 110˚C matched the results of Foreman et al. [6] (91.88 kJ/mol @ 67˚C) nicely, but the E_a for the 40˚C data did not.

Table 3-3: Factors for Shifting Creep Data to 23˚C

Temperature (˚C)	Acceleration Factor a_T	Activation Energy (kJ/mol)
23	(1)	-
40	72.97	194.58
80	438.72	92.81
110	23082	108.94

The assembled Relaxation Master Curve is shown in **Figure 3-7**. Surprisingly, the exact same test lengths and temperatures that allow predictions of more than 20 years for Creep only allow predictions of approximately 2 years in Relaxation. It is believed that the reason for this difference is because the Creep Test inputs more work into the test sample for a given time and temperature than its Relaxation counterpart that starts at the equivalent stress-strain state.

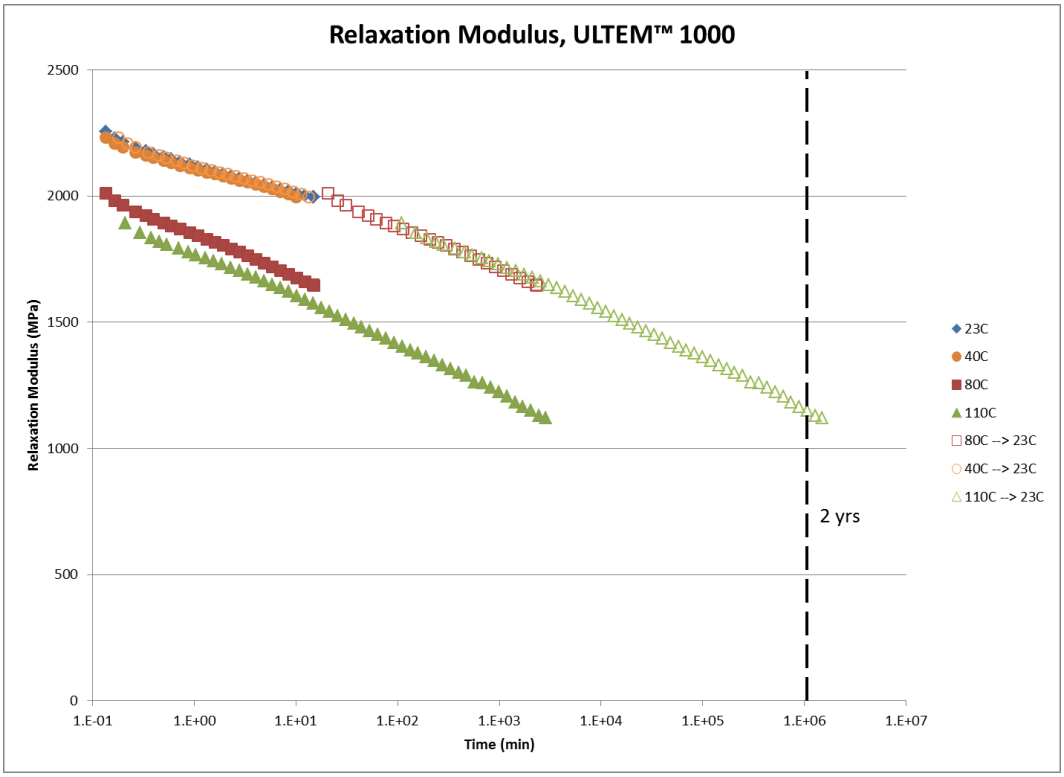


Figure 3-7: Relaxation Modulus Results and TTSP Master Curve

Table 3-4 shows the acceleration factors and calculated activation energy used to assemble the Relaxation Master Curve shown in **Figure 3-7**. Note that all activation energies are much lower than any Creep data.

Table 3-4: Factors for Shifting Relaxation Data to 23˚C

Temperature (˚C)	Acceleration Factor a_T	Activation Energy (kJ/mol)
23	(1)	-
40	1.33	12.94
80	154.65	76.90
110	519	67.80

This relaxation data will not allow predictions to 20 years at 23˚C. With the data in **Table 3-4** and the test planning methodology discussed in section 3.1, it would be relatively simple to plan a test that would allow predictions farther into the future, at temperatures higher than 23˚C, or both. For planning a Relaxation test with PEI, using an activation energy of approximately 70 kJ/mol is recommended.

Chapter 4

Analytical Solutions

4.1 Elasticitiy

As briefly explained in Chapter 2, the elasticity solution is used to find the viscoelasticity function. Although well covered by literature, this exercise was repeated for thorough understanding.

The expressions for stress distributions and associated deformations were derived by G. Lamé in 1833 and are very well covered by Timoshenko and Goodier [8] as well as Ugural and Fenster [3]. This classic thick walled cylinder problem starts with a definition similar to **Figure 1-2**, with a and b referring to the inner and outer radii while p_o and p_i stand for the outer and inner pressures on those surfaces. **Equation 4-1** can be arrived at via manipulation of the Compatibility Equation or by use of Airy's Stress Function converted for polar coordinates. To use Airy's Stress function, observation of axisymmetry is key in choosing the appropriate functions.

Equation 4-1: General Solution to the Thick Walled Cylinder Problem

$$\Phi = A\log r = Br^2\log r + Cr^2 + D$$

From here, the general equations for the stresses in the r and θ directions are shown in **Equation 4-2** by taking the appropriate partial derivations for polar coordinates.

Equation 4-2: General Stress and Displacement Solutions

for the Thick Walled Cylinder

$$a) \qquad \sigma_r = \frac{1}{r}\frac{\delta \Phi}{\delta r} = \frac{A}{r^2} + B(1 + 2\log r) + 2C$$

$$b) \qquad \sigma_\theta = \frac{\delta^2 \Phi}{\delta r^2} = -\frac{A}{r^2} + B(3 + 2\log r) + 2C$$

$$c) \qquad \sigma_{r\theta} = 0$$

$$d) \qquad 2Gu_r = -\frac{A}{r} + B[(\kappa - 1)r\log r - r] + C(\kappa - 1)r$$

$$e) \qquad 2Gu_\theta = B(\kappa + 1)r\theta$$

Equation 4-1 and **Equation 4-2** were also crosschecked with the work by Timoshenko and Goodier [8], which has a much more detailed discussion. **Equation 4-2d** above, the B term causes a θ direction displacement that is dependent on θ itself. Since the problem is axisymmetric in nature, B must be equal to zero. **Equation 4-3** shows the boundary conditions to be applied.

Equation 4-3: Boundary Conditions for Elastic Solution

$$\sigma_r(r = a) = 0$$

$$u_r(r = b) = u_0$$

Solving the general radial stress equation with the applied boundary conditions yields the equations for stresses in the r and θ directions shown in **Equation 4-4**.

Equation 4-4: Stresses in the Thick Wall Cylinder with Constant Deformation (G,κ)

$$\sigma_r = \frac{4u_0 Gb}{2a^2 + (\kappa - 1)b^2}\left(1 - \frac{a^2}{r^2}\right)$$

$$\sigma_\theta = \frac{4u_0 Gb}{2a^2 + (\kappa - 1)b^2}\left(1 + \frac{a^2}{r^2}\right)$$

By converting G and κ to E and ν, the stresses in **Equation 4-4** can be shown to be equivalent to the stress functions found by algebraic manipulation of the original Lamé functions, shown in **Equation 4-5**.

Equation 4-5: Stresses in the Thick Wall Cylinder with Constant Deformation (E,ν)

$$\sigma_r = \frac{-u_0 E}{\left[(1-\nu)r + (1+\nu)\frac{a^2}{r}\right]}\left(\frac{a^2}{r^2} - 1\right)$$

$$\sigma_\theta = \frac{u_0 E}{\left[(1-\nu)r + (1+\nu)\frac{a^2}{r}\right]}\left(\frac{a^2}{r^2} + 1\right)$$

While the stress in the θ direction is not the primary focus of this work, it is worth noting that any failure, yielding, other plasticity, viscoplasticity, or embrittlement effects will initiate at the inner radius and will be driven by stresses in the θ direction. **Figure 4-1** shows non-dimensionalized stresses in the r and θ direction as well as the equivalent Von Mises stress (as compared to the yield strength for failure prediction). In elasticity, constant deformation and constant stress in the radial direction are equivalent to each other.

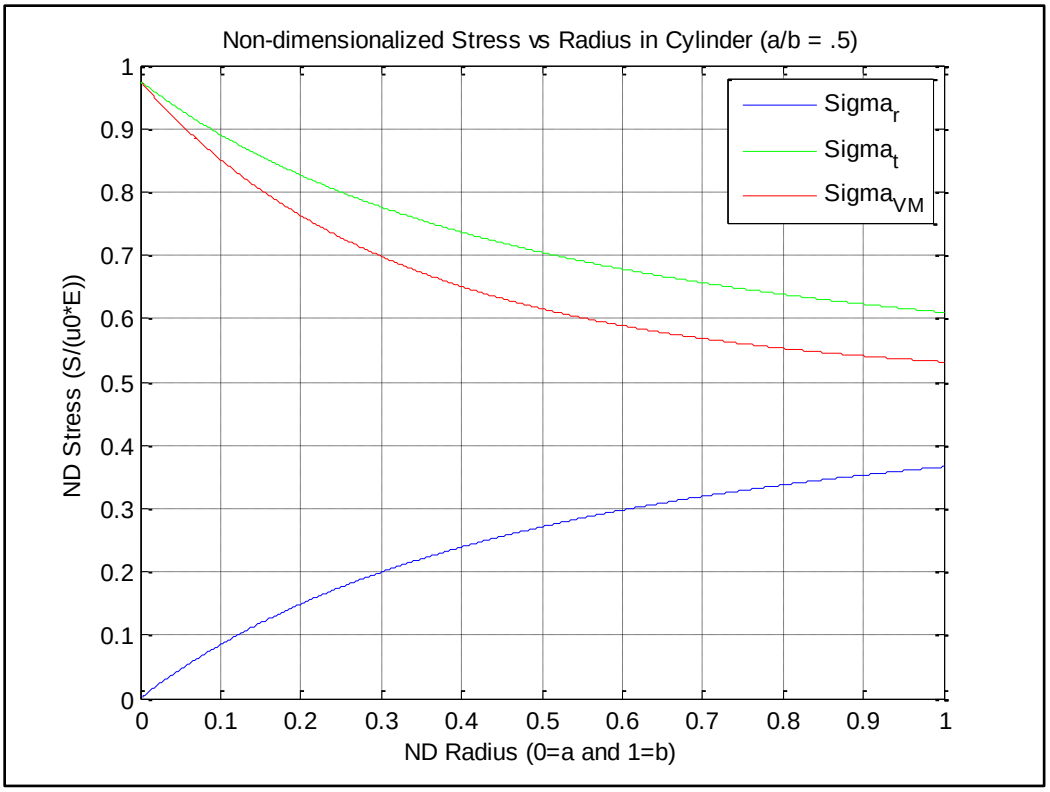


Figure 4-1: Non-dimensionalized Stresses in a Thick Walled Cylinder with Constant Radial Deformation

4.2 Viscoelasticity

One with a strong background in elasticity and new to viscoelasticity might see examples such as the constant stress cylinder presented by Christenson [5] and attempt to manipulate the solution in ways similar to those demonstrated in section 4.1. Upon re-evaluation of the problem statement diagram in **Figure 1-2** and comparison with the illustrations in **Figure 2-1**, it becomes clear that these problems are as dissimilar as the Creep and Relaxation tests themselves. In this section, a function will be fit to the Master Curve and then the complex modulus will be combined with the Elasticity solution.

4.2.1 Prony Series Approximation for Creep

To fit a function to the assembled Master Curve data from section 3.5, Prony Series approximation was used. A short program was written and can be seen in Appendix C. The intention of this program is to fit a finite number of coefficients to **Equation 4-6**. The Prony Series representation of D(t) is effectively a generalized Voigt-Kelvin mechanical model, wherein a spring element is in series with infinite Voigt-Kelvin elements.

Equation 4-6: Prony Series Representation for Creep Compliance [9]

$$D(t) = \frac{1}{E_0} + \sum_{i=1}^{n} \frac{1}{E_i}\left(1 - e^{-t/\tau_i}\right)$$

The program was written to utilize a least squares approach as described by Brinson and Brinson [4]. Sign control methodology was added as detailed by Bradshaw and Brinson [9] to force all moduli to be positive in matching with physics. As recommended by Bradshaw and Brinson [9] , the time constants were chosen logarithmically spaced, with the minimum and maximum time constants outside the dataset range. Instead of scaling the range by an arbitrary constant, the time constants were chosen at each decade and ranged to 1 decade outside the available test data. Refer to **Table 4-1**.

Unfortunately, the standard least squares fit routine built into Matlab® version R2013b was not able to simultaneously fit this many constants. Fits with 3 and 5 constants had very poor R^2 values. Instead, the program returns the same values for E_i that were originally input, but will also give goodness-of-fit calculations that can be used for a manual fitting routine.

As a side note, investigation using a later version of Matlab had a different type of fitting routine. The code in Appendix C experienced trouble in Matlab® version R2014a. The alternate methods for the later versions were not tried, but may offer improvements over this code.

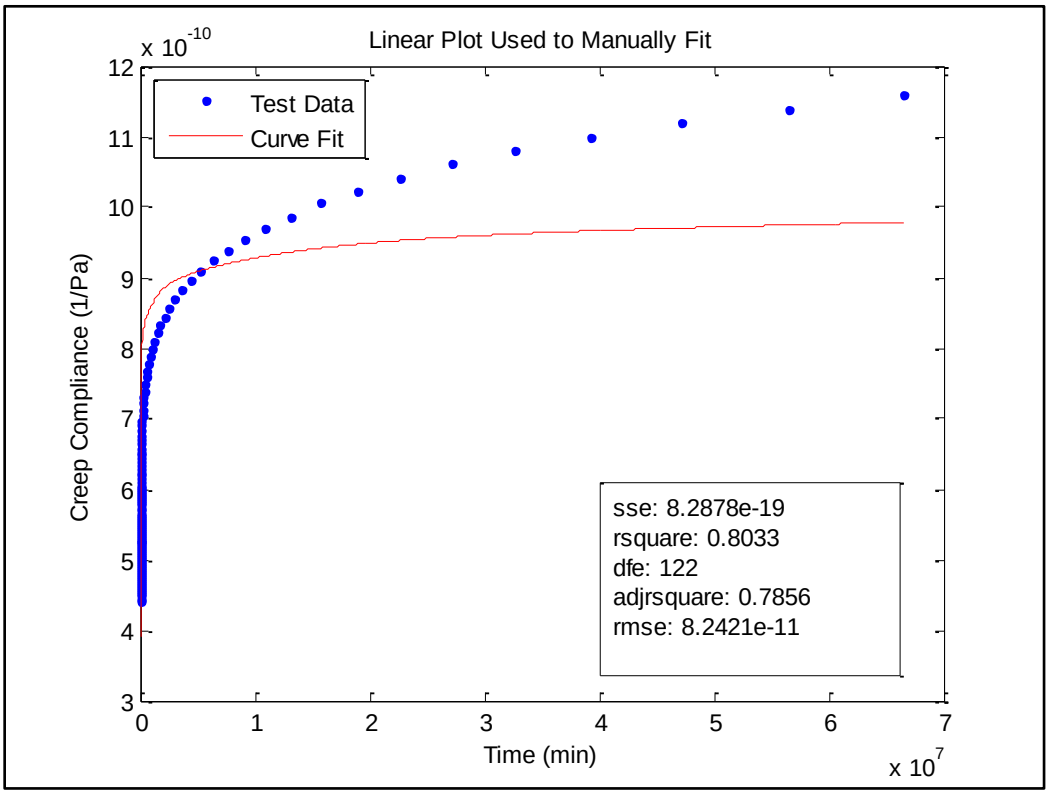


Figure 4-2: First Step in Manual Prony Series Fit (All E_i = 1.5e10)

The simple manually fitting routine that was developed starts by using the Young's Modulus as reported by the material manufacturer or from a standard tensile test for E_0. All other E_i terms are set equal to each other and the fit is plotted with the test data as seen in **Figure 4-2**. All E_i terms are scaled up and down equally until both curves are of the same order of magnitude. **Figure 4-2** shows the output from this rough

optimization with E_0 = E and E_i = 1.5e10. Next, the scale is adjusted to view the comparison of the two curves over the first decade of data. The first E_i is adjusted, noting inverse sensitivity with the curve fit, until the first few points are roughly equal. The scale is then adjusted to show the next decade of information and the next E_i term is adjusted. It was found that rechecking the previous E_i term before moving forward will save time and allow fine tuning as needed. The final result of this routine with this data (r2 = .9999) is shown in **Figure 4-3**.

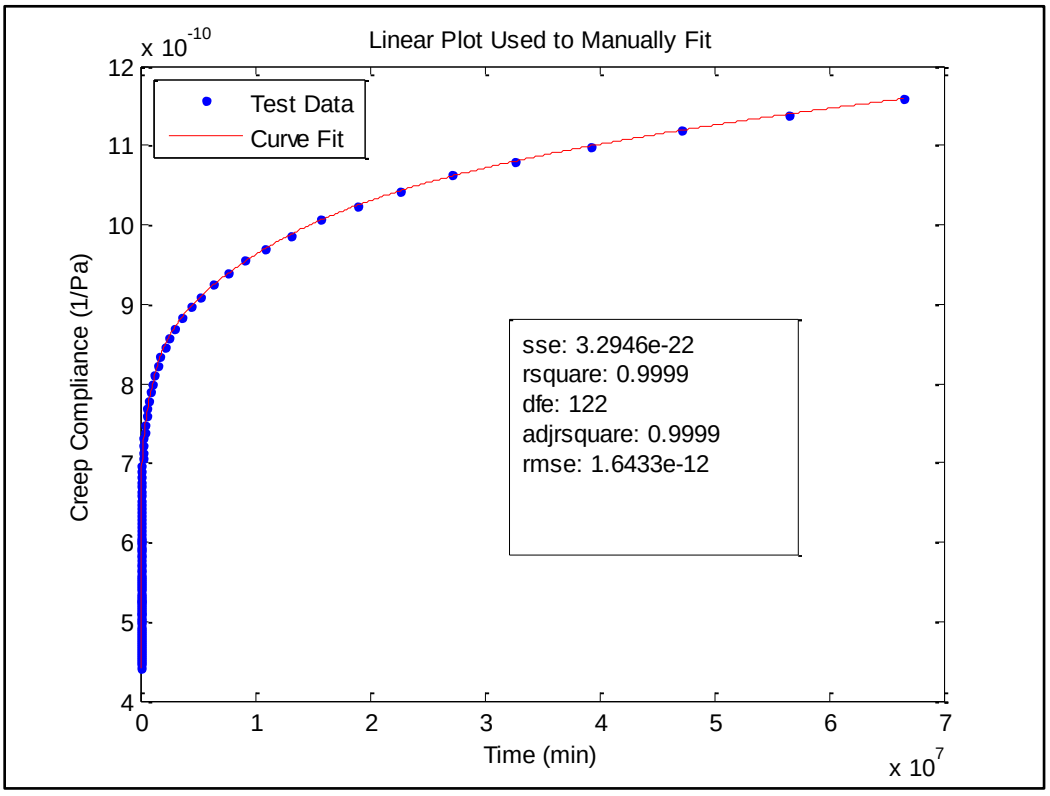


Figure 4-3: Plot Used for Manual Fitting of Creep Prony Coefficients

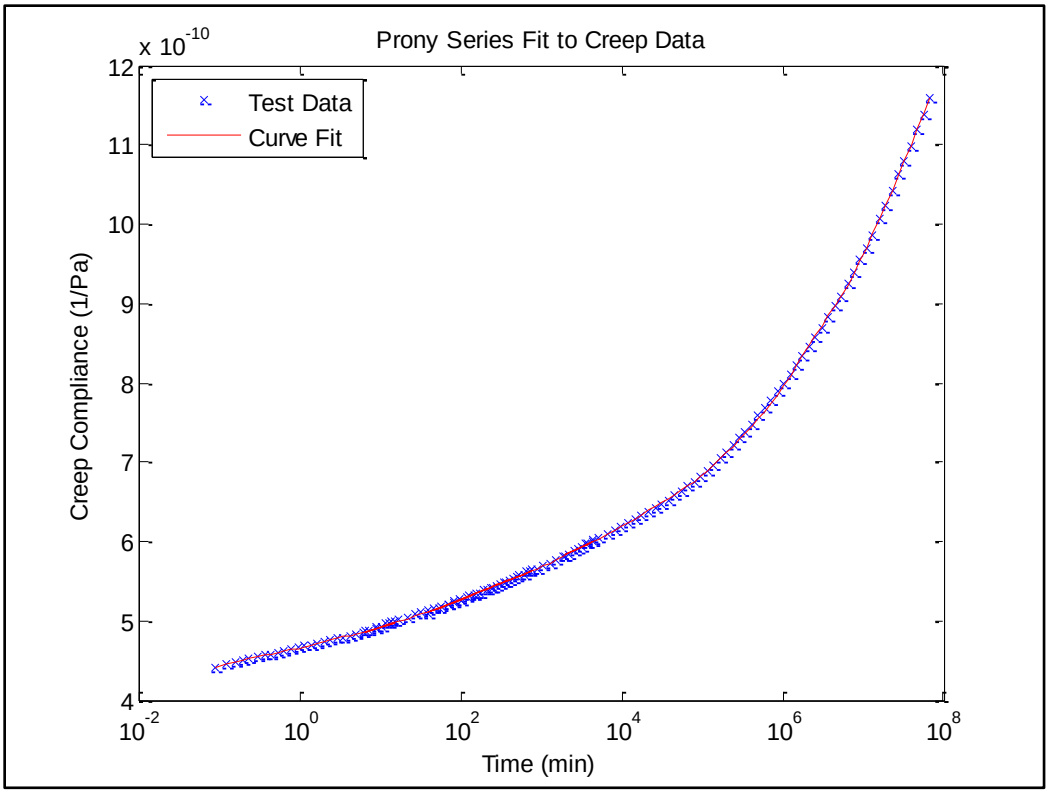


Figure 4-4: Log Plot of Creep Prony Series Fit

Figure 4-4 shows the same curve fit data in a log plot similar to all the plots from Chapter 3. **Figure 4-5** and **Table 4-1** show the coefficients used to achieve this Prony series fit. As discussed by Brinson and Brinson [4], the curvature seen in **Figure 4-5** smoothed as the fit was refined. Although the Prony Series approximation presented here has time coefficients ranging from 1e-2 to 1e8 minutes, the model is only valid within the range of the assembled Master Curve. This representation is calculated to be valid from 2 minutes to approximately 120 years, but due to the wide spacing of data at long times and the sensitivity to the linear interpolation methods for TTSP used, it is recommended to limit the use of this approximation to 20-30 years. Collecting more datapoints from short tests at low and high temperature would reinforce this model.

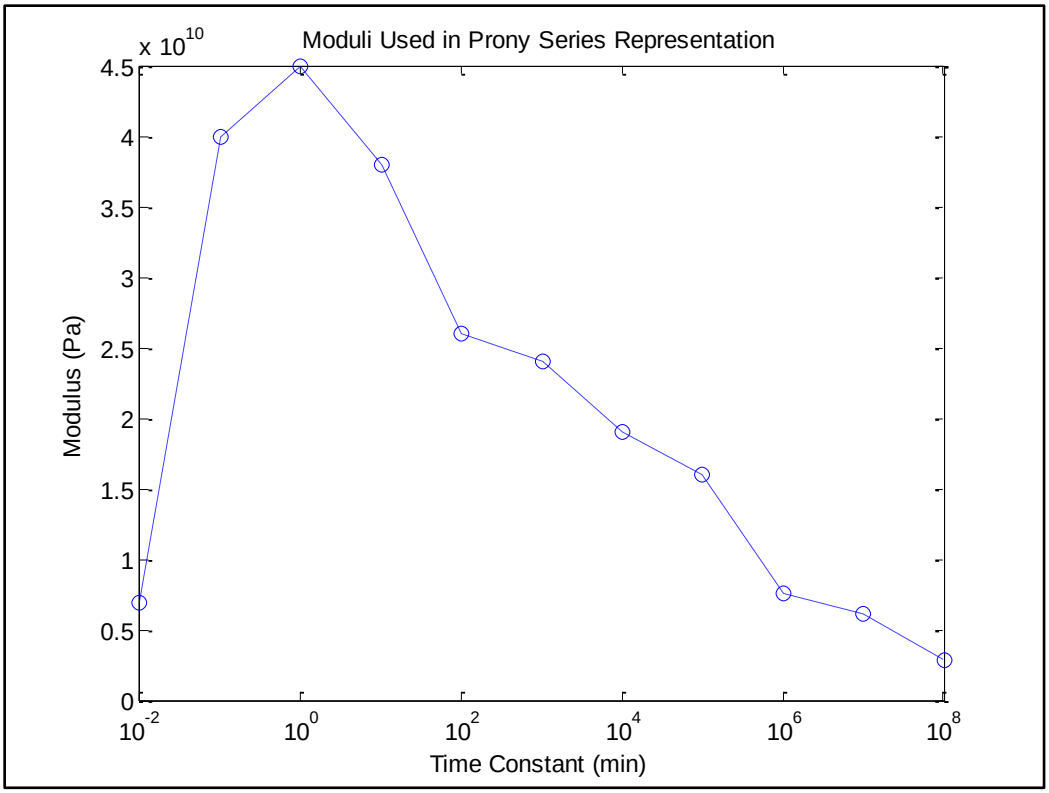


Figure 4-5: Prony Series Coefficients for D(t) of PEI

Table 4-1: Time Constants and Moduli for Prony Series Representation of D(t) of PEI

Time Constant τ (min)	Modulus E (MPa)
(N/A)	3580
1E-02	6900
1E-01	40000
1E+00	45000
1E+01	38000
1E+02	26000
1E+03	24000
1E+04	19000
1E+05	16000
1E+06	7600
1E+07	6100
1E+08	2850

4.2.2 Prony Series Approximation for Relaxation

Techniques very similar to those described in section 4.2.1 were also employed for the Relaxation Master Curve. Prony Series approximation for Relaxation has a different form from Creep, as seen in **Equation 4-7**. The Prony Series representation of E(t) is mathematically the Wiechert Model, with one spring in parallel with infinite Maxwell elements (generalized Maxwell model with one additional spring in parallel). **Figure 4-6** shows the results of this curve fit.

Equation 4-7: Prony Series Representation for Relaxation Modulus [9]

$$E(t) = E_\infty + \sum_{i=1}^{n} E_i e^{-t/\tau_i}$$

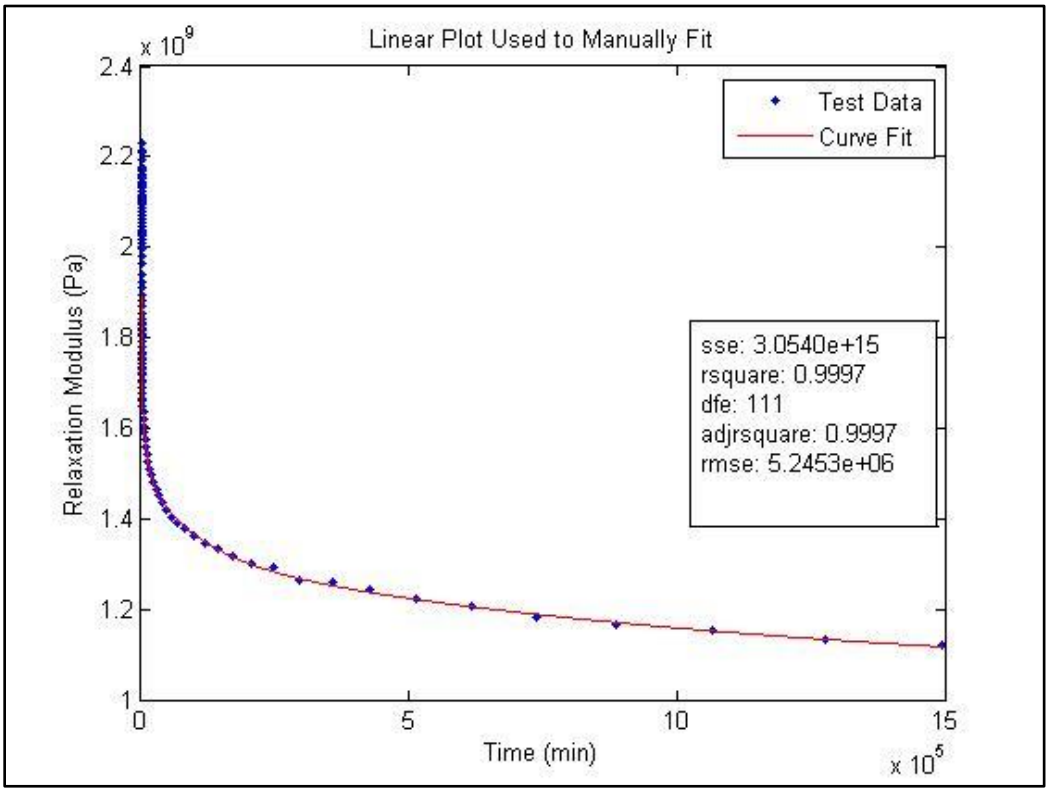


Figure 4-6: Plot Used for Manual Fitting of Relaxation Prony Coefficients

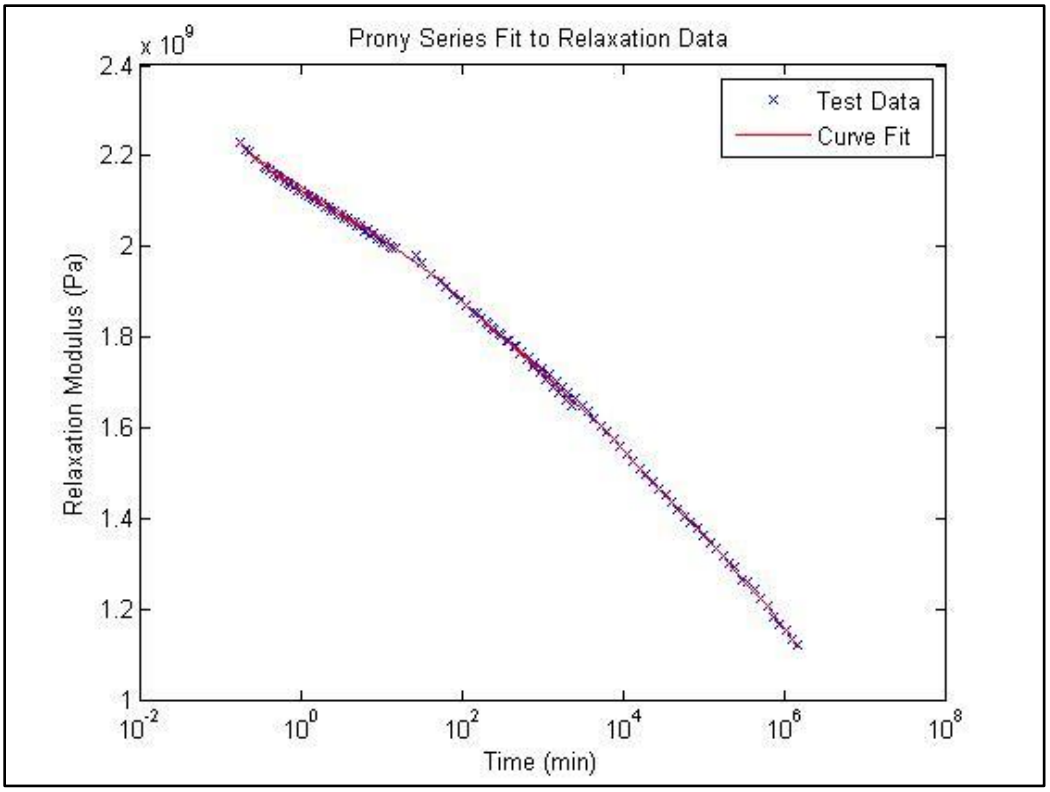


Figure 4-7: Log Plot of Relaxation Prony Series Fit

Figure 4-7 shows the same Prony Series fit for Relaxation Modulus in the more familiar log plot form. It is also worth noting that the fit function in Matlab® version R2013b did automatically optimize the Moduli used in the fit. **Figure 4-8** and

Table 4-2 show the coefficients both graphically and in tabular format. The lack of smoothness in the curve in **Figure 4-8** indicates that the fit could be refined by adding more terms, but as indicated in **Figure 4-6**, the r^2 value was .9997, indicating a good enough fit for the scope of this work.

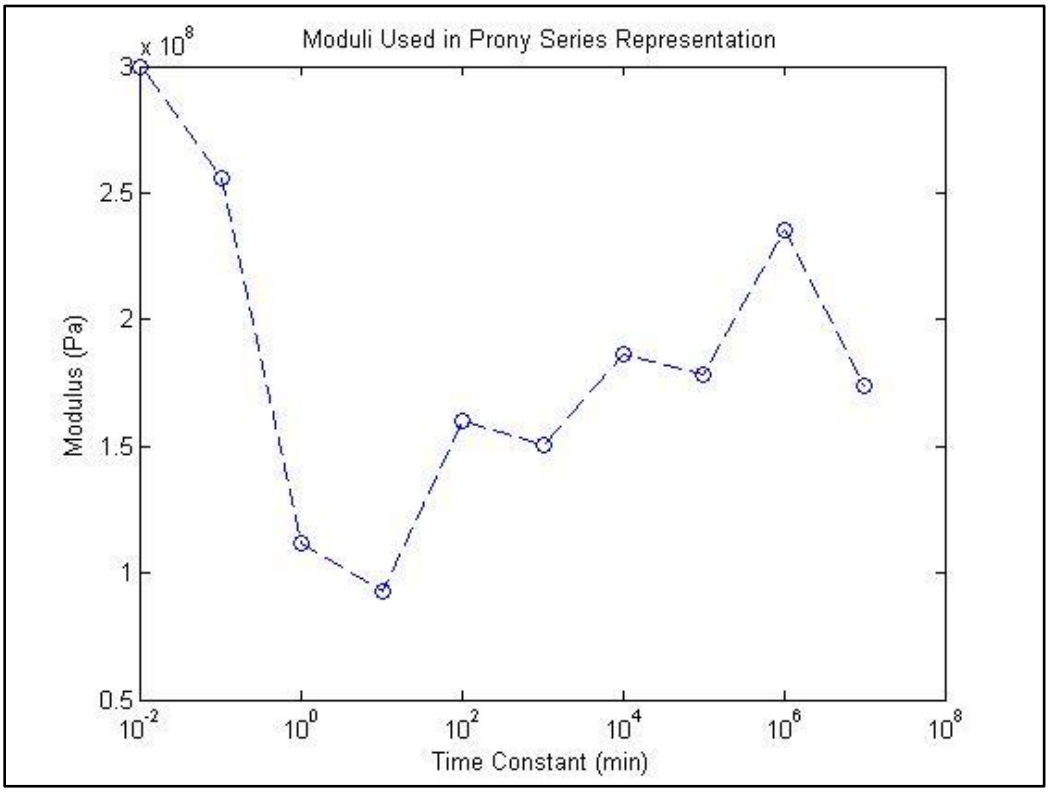


Figure 4-8: Prony Series Coefficients for E(t) of PEI

Table 4-2: Time Constants and Moduli for Prony Series Representation of E(t) of PEI

Time Constant τ (min)	Modulus E (MPa)
1E-02	300
1E-01	255.6
1E+00	111.9
1E+01	92.4
1E+02	160
1E+03	150.8
1E+04	186
1E+05	178.3
1E+06	235.2
1E+07	174.2
(Inf)	914.5

4.2.3 Utilizing the Elastic Solution

To convert **Equation 4-5** for use in viscoelasticity, the elastic modulus and Poisson's Ratio must be replaced with their complex counterparts. This conversion is shown for clarity in **Equation 4-8**.

Equation 4-8: Radial Stress Distribution in the Laplace Domain

$$\overline{\sigma_r} = \frac{-u_0 s\bar{E}}{\left[(1-s\bar{\nu})b + (1+s\bar{\nu})\frac{a^2}{b}\right]}\left(\frac{a^2}{r^2}-1\right)$$

For this work, the Poisson's Ratio will be assumed to be constant. As discussed by Brinson and Brinson [4], assuming a constant Poisson's Ratio is effectively assuming that the Shear Modulus G and Bulk Modulus K scale equally through time (K(t)/G(t) = constant). If v(t) is constant, then the Laplace transform of v_0 is v_0/s. Substitution and rearrangement of the terms in **Equation 4-5** yields **Equation 4-9**.

Equation 4-9: Radial Stress Distribution in Laplace Domain with Constant v

$$\overline{\sigma_r} = \frac{-\bar{u}\bar{E}}{\left[(1-\nu_0)b + (1+\nu_0)\frac{a^2}{b}\right]}\left(\frac{a^2}{r^2} - 1\right) = C\bar{u}s\bar{E} = \frac{C\bar{u}}{s\bar{D}}$$

According to **Equation 4-9**, if the Laplace inverses of the right hand terms can be found, then that solution can be multiplied by a constant representing all of the geometric effects. The Laplace transforms and their inverses are covered in section 4.2.4.

4.2.4 Laplace Transforms

Because the displacement is a constant for this problem, the time dependent displacement is modeled by **Equation 4-10**. As commonly encountered, the Laplace transforms for E(t) and D(t) are easily found. Both are shown in **Equation 4-11.** Note the use of the Heaviside function H(t) to indicate an instantaneous crimp operation, when in reality this step would happen over some small time.

Equation 4-10: Laplace Transform for Constant Displacement

$$u(t) = u_0 H(t) \;\; and \;\; \bar{u}(s) = \frac{u_0}{s}$$

Equation 4-11: Laplace Transforms for Creep and Relaxation

$$\frac{C\bar{u}}{s\bar{D}} = \frac{Cu_0}{s^2\bar{D}} = \frac{Cu_0}{\frac{s}{E_0} + \sum_{i=1}^{n}\frac{s}{E_i} - \sum_{i=1}^{n}\frac{s^2/E_i}{s + {}^{1}/_{\tau_i}}}$$

$$C\bar{u}s\bar{E} = C\frac{u_0}{s}s\bar{E} = Cu_0\bar{E}$$

Matlab was once again employed (see Appendix D) to symbolically solve the inverse of the compliance function in **Equation 4-11**. The Creep function code for 1 or 2 parameters can be executed in less than a minute on a typical laptop. A solution to the 3 parameter inverse failed using a typical laptop (out of memory), took approximately 40 minutes to execute on a modern multiprocessor workstation with excess memory, and is approximately 4.5 pages long. All attempts to solve the 11 parameter equation failed. It is clear from this exercise that obtaining the data that fits the application best is much easier than converting types.

The solution for time dependent stress using the Relaxation Modulus is relatively straightforward for this set of assumptions, and is shown in **Equation 4-12**. It's worth noting that for this simple static case, the viscoelastic solution is effectively the elastic solution with the Relaxation Modulus substituted for Young's Modulus. Brinson and Brinson [4] covered a similar example of an axially loaded bar and came to a similar conclusion for the simplified static case by using the convolution integral.

Equation 4-12: Inverse Laplace Solution for Relaxation Modulus

$$\mathcal{L}^{-1}\{Cu_0\bar{E}\} = Cu_0E(t) = \frac{-u_0E(t)}{\left[(1-\nu_0)b + (1+\nu_0)\frac{a^2}{b}\right]}\left(\frac{a^2}{r^2} - 1\right) = \sigma(t)$$

4.3 Results

All of the previous information was combined, and the Relaxation Modulus was re-examined with respect to the Young's Modulus. This is shown graphically in **Figure 4-9**.

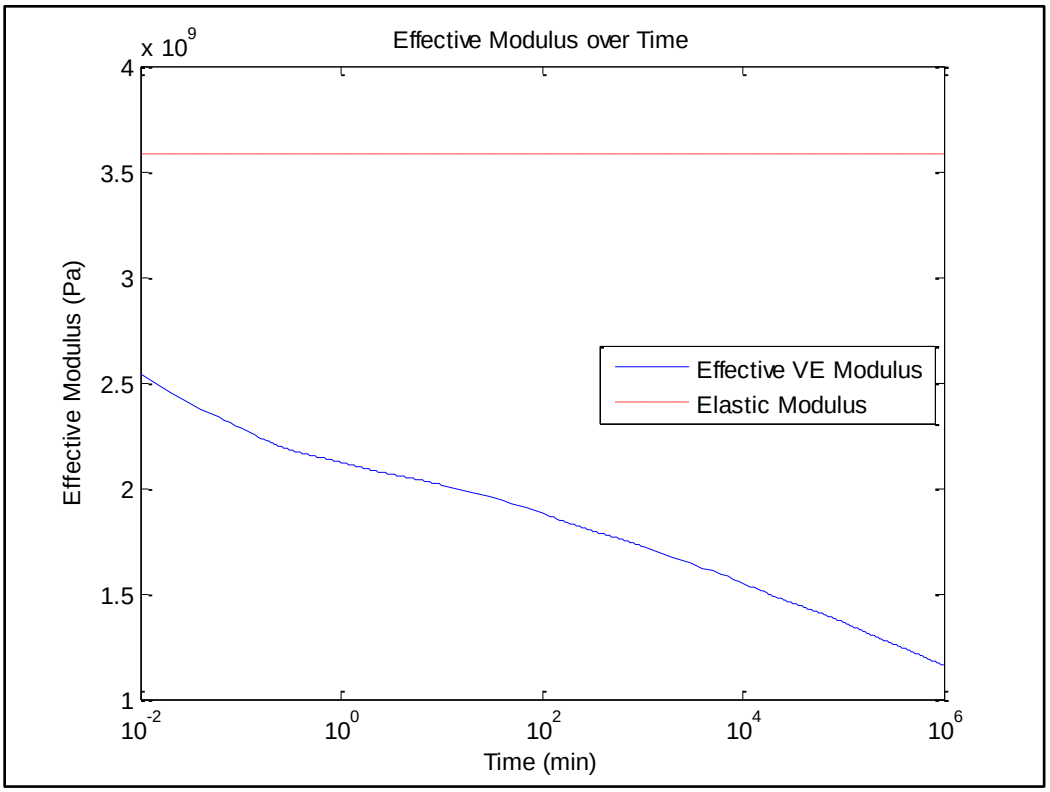


Figure 4-9: Comparison between Relaxation and Young's Modulus

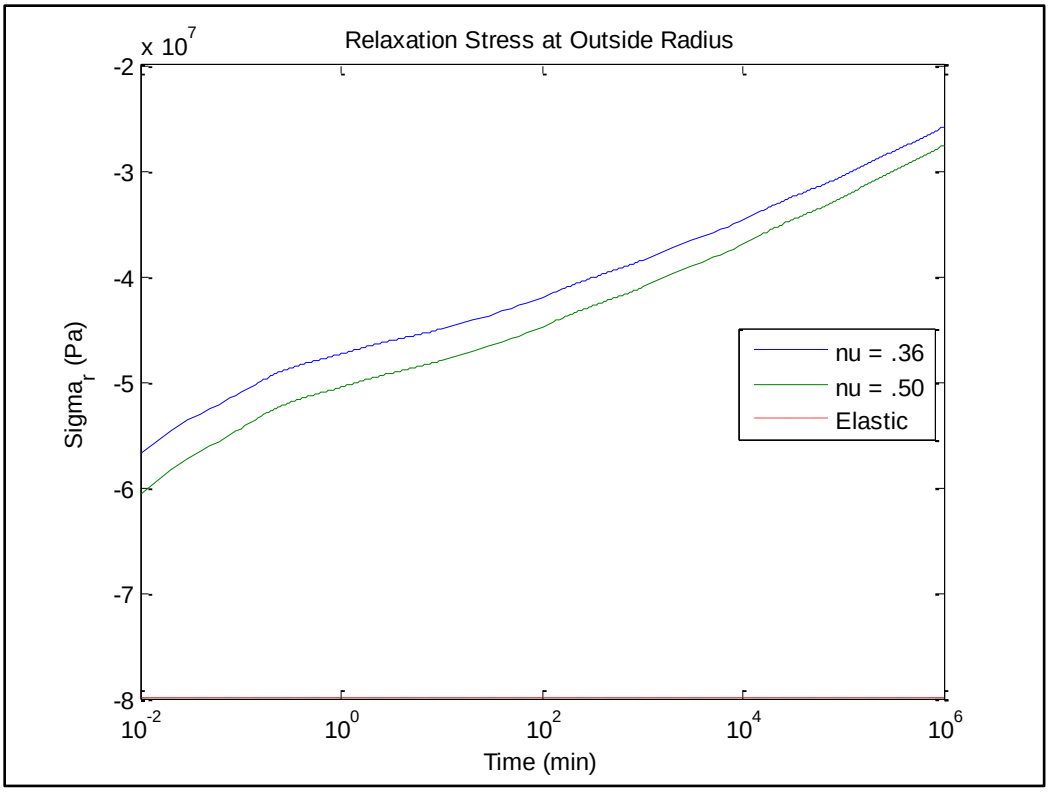


Figure 4-10: Pressure (Stress) at the Crimp Interface (outside radius)

As discussed previously, the interface pressure is equal to the stress at the outside radius of the plastic cylinder. This pressure was calculated and plotted over time in **Figure 4-10**. Note that the stress is negative because the material at the interface is in compression and the stress tends toward zero at long times due to viscoelastic relaxation.

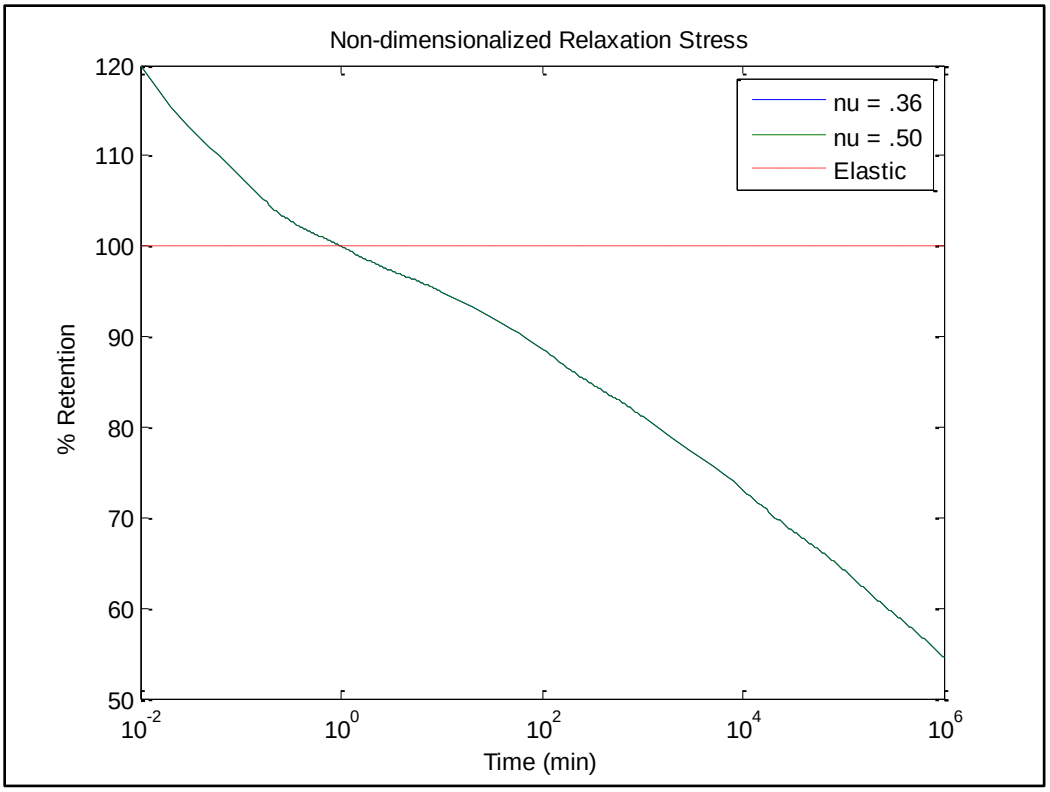


Figure 4-11: Stress Non-dimensionalized by the Stress Predicted at t = 1 minute

Last, the data was non-dimensionalized in order to view relaxation as a percentage. The data was non-dimensionalized by the elastic stress and by the stress after 1 minute of relaxation. **Figure 4-11** shows the non-dimensionalized stress as compared to the 1 minute stress. As can be seen, the stress in the neighborhood of 1 minute is very similar to that predicted by elastic theory. The 1 minute stress was chosen because it is more easily empirically tested.

In **Figure 4-11**, the stress at each Poisson's Ratio was non-dimensionalized by its own stress at 1 minute. **Figure 4-11** shows that the ring is predicted to lose nearly half of its retention strength after a little more than 2 years and reaches its specification strength of 66% in less than 6 weeks and that Poisson effects over time have very little effect.

By this non-dimensionalization technique, all geometric effects are removed. Because of that, **Figure 4-11** is effectively a plot of E(t)/E(t=1) vs time. By non-dimensionalization, the plot in **Figure 4-11** can be shown to theoretically stand for the non-dimensionalized viscoelastic stress at any point within the thickness and for any size thick wall cylinder over time.

Chapter 5

Numerical Solutions

The original plan was to import the Creep or Relaxation data into a commercially available finite element analysis (FEA) package and to compare the analytical results with the numerical results. Unfortunately, commercially available software from ANSYS and Dassault Systémes (Abaqus) rely heavily on time measurements of Shear Modulus G and Bulk Modulus K. Neither package *easily* accepts standard tensile Creep or Relaxation data.

It is worth noting that ANSYS APDL does support use of E* and v* data from DMA testing. To utilize this feature, storage modulus E' and loss modulus E'' (or tanδ) data is needed over several decades of frequency. It appears that frequency temperature superposition is also supported, but this was not tested.

Both of these systems can be fitted with custom solutions, but building a custom material model would have delayed the delivery of this work.

Chapter 6

Conclusions

Although the Relaxation test data does not allow prediction to 20 years, it does predict the decay in retention force, including the point of interest when the product falls below specification. The importance of this work lies in the ability to predict the performance of the product well into the future before it ever leaves the virtual drawing board. This prediction shows a faster delay than expected, spurring higher interest in even longer term effects.

6.1 Key Learnings

Several critical pieces of information were discovered along this journey. The first is a deep appreciation for the various material tests available at this point in time. So many test methodologies are available and so much analysis is done automatically with modern equipment that it takes a good deal of study just to understand and appreciate the results. Misunderstanding of the DMA data presented earlier in this work caused approximately one month delay *even though the data was readily available*.

Along those lines, it is appropriate to acknowledge an early shortcoming. In common mechanical engineering, it is easy to become most comfortable and reliant upon Young's Modulus and Poisson's Ratio. In the world of elasticity, these two take care of a majority of applications, and easy conversion between E, v, G, K, and λ means that we can pick favorites and convert as needed. In viscoelasticity, the selection of mechanical property *type* drives the testing and form of the results, as seen by the inability to convert Creep data for this application. Even though the Creep data gives the ability to predict longer term effects, it is much more complicated to use *for this application*. Other

applications will readily use D(t) and experience troubles with any of the other time dependent material properties.

The last item to mention was the difference in activation energies between Creep and Relaxation. This really closes the section where it began – understanding the test methodology at a deep level. Upon discovery of an activation energy and reference temperature for Creep, the times and temperatures test plans for Creep and Relaxation were calculated *using the Creep information*. Due to this misunderstanding, this work is not able to predict as far into the future as requested by the original problem statement.

6.2 Future Work

6.2.1 Longer Term Prediction

As discussed in the previous section, the desire for this work was originally to predict 20 years into the future at temperatures up to 40°C. With the Relaxation test better understood, it would not be a major work to put together a second Relaxation test plan to achieve those results.

6.2.2 Internal Pressure

If the elasticity solution was solved including internal pressure, then this solution could be better applied to hydraulic and pneumatic applications. For static internal pressure applications, the Laplace inversion in section 4.2.4 could be directly applied. This would also allow the effects of cyclical internal pressures to be examined, but the Laplace transformations and inversions would need to be repeated for p(t).

6.2.3 Direct Testing

An original desire for this work was to confirm the results with some additional empirical testing of the application itself. An axial pull test could be assembled to show correlation between theoretical predictions and reality.

6.2.4 Local Effects

In reality, local effects are commonly used in crimping. Localized radial deformations increase axial retention forces dramatically. When parts fail, the location is commonly at the edge where the very stiff outer crimp band terminates. Both of these highly important effects are not covered by this work. Due to the complexity of these topics, it is believed that advanced numerical (finite element) models will be needed.

6.2.5 Viscoplasticity

In a manufacturing environment, process variation can cause quality concerns if not controlled. To qualify a crimping process for small components, it may be noted that initial retention forces vary more in the elastic region than if the material is crimped into the plastic region because the output stress for the additional strain is negligible. But what happens at long times?

6.2.6 Composite Effects

A common solution for injection molded applications needing higher stiffness and/or strength is to reinforce the plastic with short glass fiber. With this reinforcement comes a three dimensional distribution of anisotropic material properties. Where the manufacturer introduces the material to the mold may have as much to do with performance as the original part design.

6.3 Final Remarks

This work did not accomplish all of the goals that were originally desired, but it set the groundwork to answer those concerns. A material was analyzed, compared with data from the manufacturer, and the analysis routine was itself observed for improvements. The calculated values for others to utilize this material analysis were delivered. The operations to apply this material model to the thick walled cylinder with constant deformation were discussed and the results were shown, enabling others to apply some of this information to other applications. The only deliverable that was missed was the prediction to at least 20 years at 23°C, but the goal to predict time to specification pull strength was met and all the information needed to plan an effective test to get 20-year data is contained herein. Last, a non-dimensionalized model that theoretically predicts the viscoelastic stresses at any point in a PEI thick wall cylinder of any size with constant radial deformation was presented, showing a 1/3 drop in retention strength within 6 weeks and a drop of nearly ½ after 2 years at 23°C.

Appendix A

Drawing for Test Samples

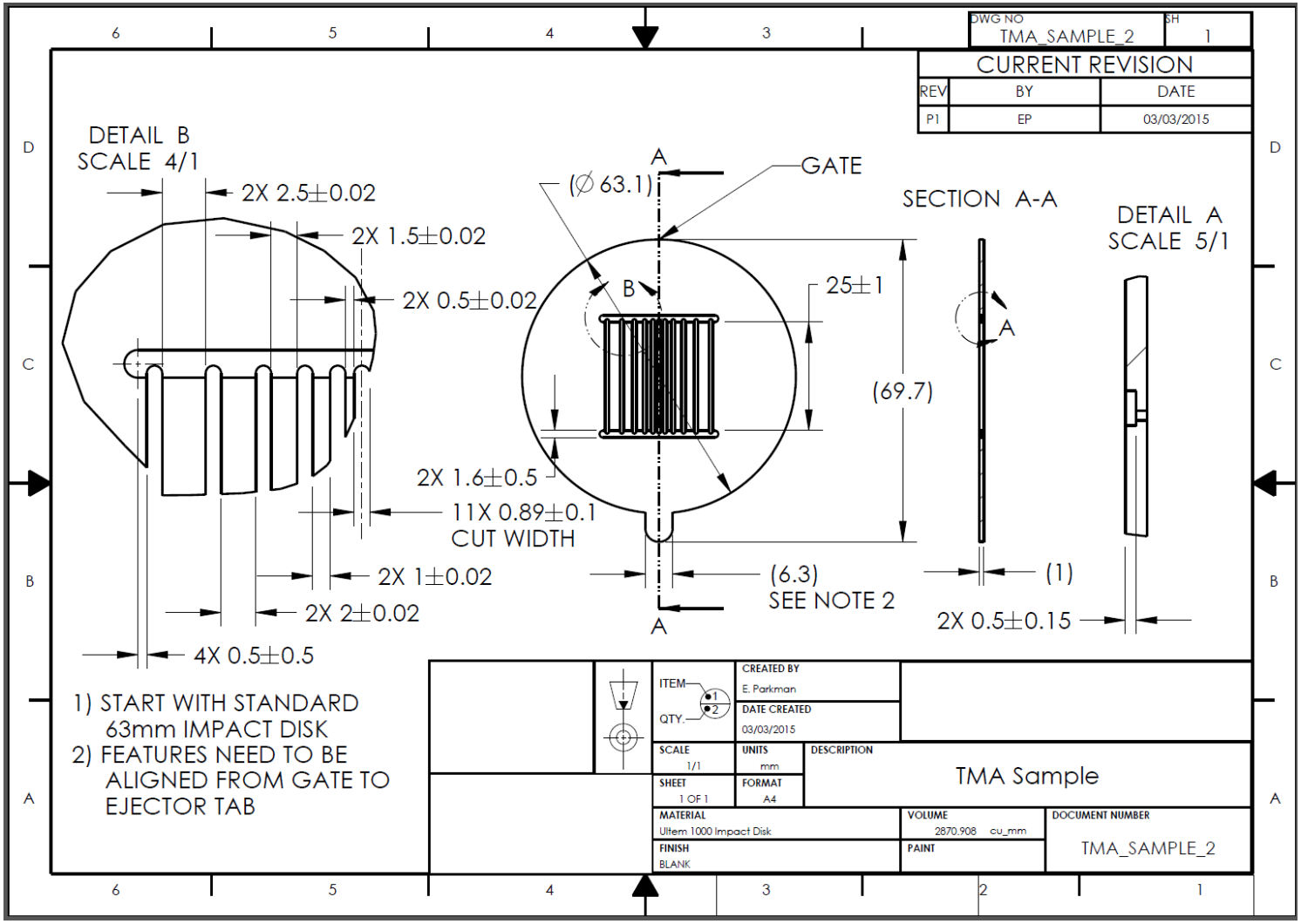


Figure A-1: Drawing for Test Samples.

Appendix B

Linear Interpolations for Acceleration Factors

```
%Linear_Interpolation.m
%By Eddie Parkman
%MSME Student, University of Texas at Arlington
%Dr. Ashfaq Adnan, Advising Professor

clear

%This program linearly interpolates needed acceleration factor needed
    %for higher temp curves to line up with the previous temp curve
    %Point 0 is placed between 1 and 2.

%Input all the times and Compliances/Moduli as an array, then run all
%calculations serially
%First 3 are for Creep, 0 to separate, Last 3 are for Relaxation.

t1 = [  5.9195
        54.91256
        3123.531
        0
        .334166
        14.80017
        206.326392];
D1 = [  485.9186
        517.8164
        593.7625
        0
        2178.299
        1996.282
        1829.728];

t2 = [  7.11967
        67.07476
        3752.359
        0
        .400833
        15.00016
        247.591608];
D2 = [  488.2282
        520.632
        598.3316
        0
        2167.386
        1995.849
        1817.534];

t0 = [  .085834
        .1525
        .1525
        0
        .2675
        .2675
```

```
        .459166];
D0 = [  486.5808
        520.5926
        596.6436
        0
        2174.757
        1938.759
        1820.231];

%Loop through the calcs
for i = 1:length(t1)

    t_calc(i) = t1(i) + (D0(i) - D1(i))*(t2(i) - t1(i))/(D2(i) - D1(i));
    aT(i) = t_calc(i)/t0(i);

end

aT
```

```
aT =

   1.0e+04 *

    0.0073    0.0439    2.3082       NaN    0.0001    0.0155    0.0519
```

Published with MATLAB® R2013b

Appendix C

Prony Series for Creep

```
%Master_Curve_Creep.m
%By Eddie Parkman
%MSME Student, University of Texas at Arlington
%Dr. Ashfaq Adnan, Advising Professor

clear

%Input Shifted Test Data by Original Temp
Data_23 = [ 0.085834       439.9492
            0.119167       444.5102
            0.1525      447.4824
            0.185833       449.7602
            0.219166       451.607
            0.285832       454.3812
            0.3525      456.5306
            0.419166       458.3008
            0.519166       460.4355
            0.619168       462.0984
            0.7525      464.0085
            0.919166       466.0088
            1.119168       467.978
            1.3525      469.8692
            1.619332       471.6274
            1.952666       473.5435
            2.352668       475.499
            2.819334       477.4022
            3.386       479.3058
            4.086166       481.44
            4.919501       483.6121
            5.9195      485.9186
            7.11967     488.2282
            8.553       490.7148
            10.2865     493.4102
            12.35167       496.1006
            14.8185     499.0798
            14.98517       499.2644];

Data_40 = [ 6.263607399    486.5808
            8.696033075    492.3194
            11.12845875    496.1266
            13.56088443    498.898
            15.9933101     501.2393
            20.85816145    504.9072
            25.72315875    507.6726
            30.5880101     509.9853
            37.8853601     512.8914
            45.18285605    515.1882
            54.91255875    517.8164
            67.0747601     520.632
            81.66960605    523.4959
            98.69665875    526.226
```

```
            118.1683237    529.0061
            142.4928724    531.9762
            171.6824183    534.9744
            205.7366696    538.1713
            247.088271     541.5826
            298.1818346    545.1824
            358.9932062    549.0268
            431.9666333    552.9013
            519.5472387    557.149
            624.1423455    561.4778
            728.7498578    565.3276];

Data_80 = [ 37.65696373    508.6378
            52.28076749    515.8608
            66.90457125    520.5926
            81.52837501    524.2074
            96.15217877    527.13
            125.3997863    531.6894
            154.6482713    535.187
            183.8958788    538.1074
            227.7677288    541.7307
            271.6404562    544.6438
            330.1356713    548.038
            403.2551288    551.3838
            490.9997062    554.8897
            593.3667713    558.3302
            710.430906     561.6566
            856.6706985    565.1864
            1032.158976    568.8534
            1236.893983    572.5772
            1485.500841    576.4226
            1792.676618    580.5289
            2158.276099    584.7679
            2596.994161    589.1675
            3123.530943    593.7625
            3752.359331    598.3316
            4381.2623      602.3797];

Data_110 = [    1981.220388         582.1805
                2750.612694         590.8518
                3520.005   596.6436
                4289.397306         601.0408
                5058.789612         604.6435
                6597.574224         610.4559
                8136.405   615.061
                9675.189612         618.8059
                11983.38961         623.6533
                14291.63578         627.7462
                17369.205  632.4307
                21216.18961         637.2422
                25832.63578         642.3848
```

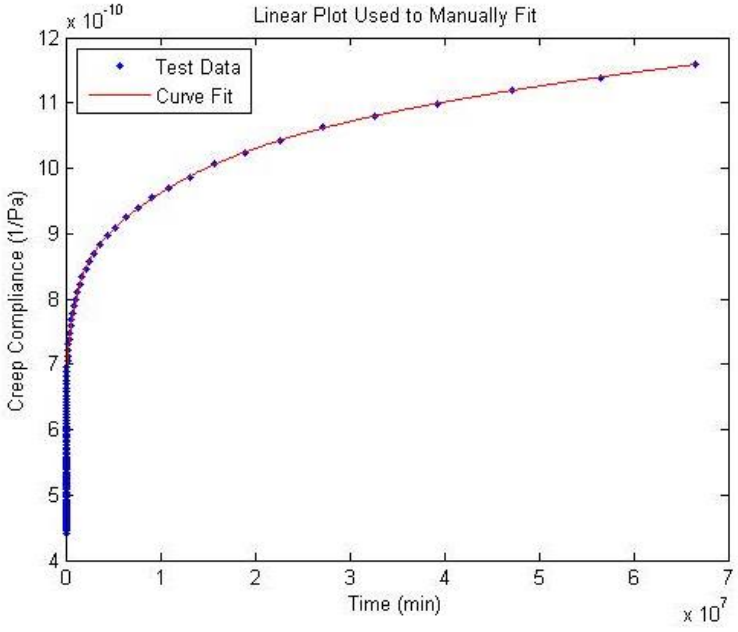
```
                31218.405  647.4106
                37377.42122         652.3765
                45071.43661         657.8697
                54304.28278         663.5909
                65075.86739         669.4141
                78155.652  675.9482
                94316.88361         682.5407
                113551.9221         689.4864
                136633.899 696.824
                164336.2229         704.5363
                197420.346 712.649
                237432.993 721.0808
                285101.2469         729.9109
                342040.617 738.9705
                410520.9871         748.3828
                492850.7879         758.0481
                591341.6819         767.992
                709837.0529         778.3349
                852145.1999         788.8784
                1022963.541         799.5098
                1227639.252         810.2859
                1473054.924         821.5697
                1767753.897         832.7821
                2121663.048         844.7162
                2546399.315         856.9668
                3055738.268         869.6588
                3667415.885         882.9201
                4401391.17 896.1837
                5281604.774         909.7368
                6337982.511         924.0216
                7605962.174         939.2694
                9127049.817         954.6051
                10952840.63         970.6253
                13143310.89         986.3844
                15772369.16         1006.486
                18926923.78         1022.705
                22712355.62         1042.026
                27254879.37         1062.536
                32706086.06         1080.394
                39247547.95         1098.587
                47096928.28         1119.107
                56516715.56         1138.88
                66475836.85         1158.549];


%Assemble all of the Data into one set
Time = [Data_23(:,1)
        Data_40(:,1)
        Data_80(:,1)
        Data_110(:,1)];  %minutes
```

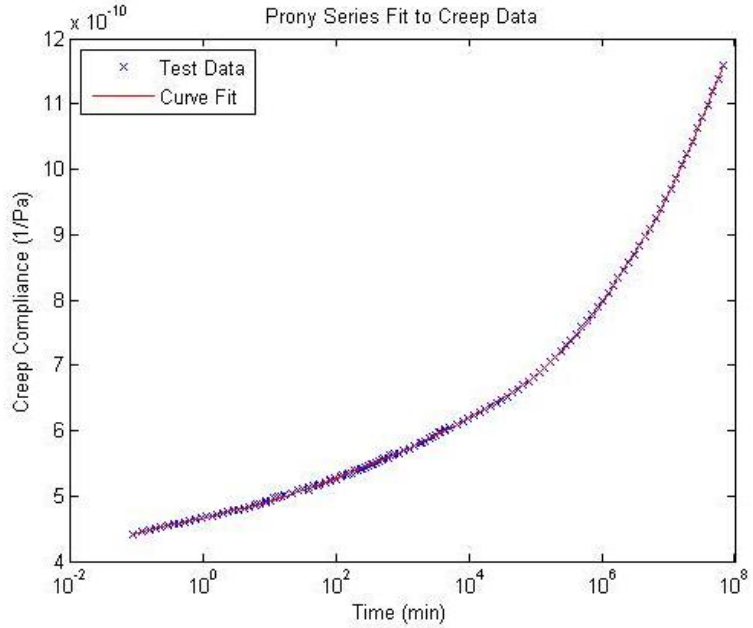
```
D_t = [Data_23(:,2)
       Data_40(:,2)
       Data_80(:,2)
       Data_110(:,2)].*1e-12;  %convert to 1/Pa

%Double Check as needed
%plot(Time, D_t)


%need to fit t = 1e-2 : 1e8
g = fittype('1/E0 + 1/E01*(1-exp(-x/1e-2))+ 1/E02*(1-exp(-x/1e-1))+ 1/E03*(1-exp(-
x/1e0))+ 1/E04*(1-exp(-x/1e1))+ 1/E05*(1-exp(-x/1e2))+ 1/E06*(1-exp(-x/1e3))+ 1/E07*
(1-exp(-x/1e4))+ 1/E08*(1-exp(-x/1e5))+ 1/E09*(1-exp(-x/1e6))+ 1/E10*(1-exp(-x/1e7))
+ 1/E11*(1-exp(-x/1e8))')

StartValues = [ 3580e6 %  E0
                6.9e9 %  E01
                4e10 %E02
                4.5e10 %E03
                3.8e10 %E04
                2.6e10 %E05
                2.4e10 %E06
                1.9e10 %E07
                1.6e10 %E08
                7.6e9 %E09
                6.1e9 %E10
                2.85e9 %E11
                ]';
Lower = zeros(1,11); %Avoid negative values
[f,gof] = fit(Time, D_t, g, 'StartPoint', StartValues, 'Lower', Lower)
Values = coeffvalues(f);

plot(f, Time, D_t)
% change the time scale to check match at each decade
% axis([0 1.1e7 3.5e-10 10e-10])
xlabel('Time (min)')
ylabel('Creep Compliance (1/Pa)')
title('Linear Plot Used to Manually Fit')
legend('Test Data', 'Curve Fit', 'Location', 'Northwest')



E(1) = Values(1);
for i = 1:(length(Values)-1)

    E(i+1) = Values(i+1);
    tau(i) = 10^(-3+i);

end

D fit = 1/E(1) + 1/E(2)*(1-exp(-Time/tau(1)))...
```

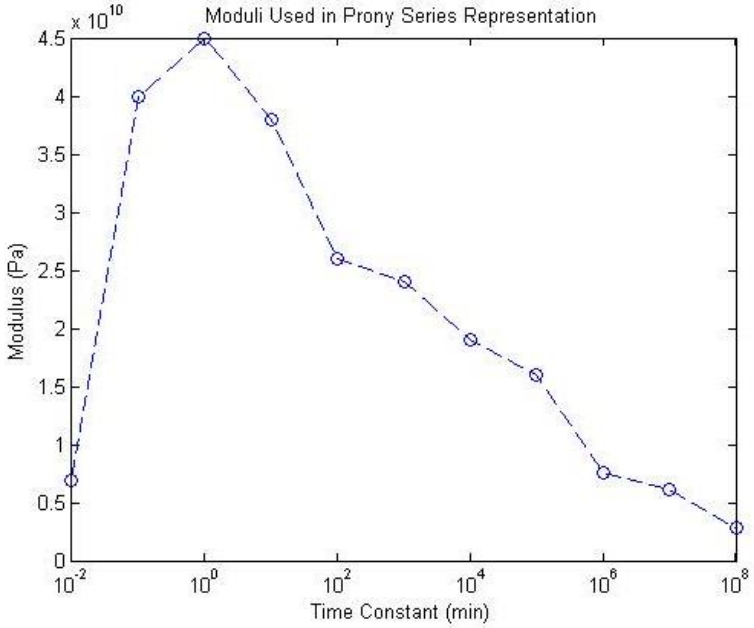
```
    + 1/E(3)*(1-exp(-Time/tau(2)))+ 1/E(4)*(1-exp(-Time/tau(3)))...
    + 1/E(5)*(1-exp(-Time/tau(4)))+ 1/E(6)*(1-exp(-Time/tau(5)))...
    + 1/E(7)*(1-exp(-Time/tau(6)))+1/E(8)*(1-exp(-Time/tau(7)))...
    + 1/E(9)*(1-exp(-Time/tau(8)))+ 1/E(10)*(1-exp(-Time/tau(9)))...
    + 1/E(11)*(1-exp(-Time/tau(10)))+ 1/E(12)*(1-exp(-Time/tau(11)));

%Make a log plot for comparision with TTSP plots
figure
semilogx(Time, D_t, 'x',Time, D_fit, 'r')
xlabel('Time (min)')
ylabel('Creep Compliance (1/Pa)')
title('Prony Series Fit to Creep Data')
legend('Test Data', 'Curve Fit', 'Location', 'Northwest')

%Plot the coefficients to evaluate for smoothness
figure
semilogx(tau,E(2:12), 'o--')
xlabel('Time Constant (min)')
ylabel('Modulus (Pa)')
title('Moduli Used in Prony Series Representation')
```

```
g =

     General model:
     g(E0,E01,E02,E03,E04,E05,E06,E07,E08,E09,E10,E11,x) = 1/E0 + 1/E01*(1-exp(-
                   x/1e-2))+ 1/E02*(1-exp(-x/1e-1))+ 1/E03*(1-exp(-x/1e0)
                   )+ 1/E04*(1-exp(-x/1e1))+ 1/E05*(1-exp(-x/1e2))+ 1/E06*(1-
                   exp(-x/1e3))+ 1/E07*(1-exp(-x/1e4))+ 1/E08*(1-exp(-
                   x/1e5))+ 1/E09*(1-exp(-x/1e6))+ 1/E10*(1-exp(-x/1e7)
                   )+ 1/E11*(1-exp(-x/1e8))

f =

     General model:
     f(x) = 1/E0 + 1/E01*(1-exp(-x/1e-2))+ 1/E02*(1-exp(-x/1e-1))+ 1/E03*(1-exp(-
                   x/1e0))+ 1/E04*(1-exp(-x/1e1))+ 1/E05*(1-exp(-x/1e2)
                   )+ 1/E06*(1-exp(-x/1e3))+ 1/E07*(1-exp(-x/1e4))+ 1/E08*(1-
                   exp(-x/1e5))+ 1/E09*(1-exp(-x/1e6))+ 1/E10*(1-exp(-
                   x/1e7))+ 1/E11*(1-exp(-x/1e8))
     Coefficients (with 95% confidence bounds):
       E0 =    3.58e+09  (-2.505e+11, 2.576e+11)
       E01 =     6.9e+09  (-9.371e+11, 9.509e+11)
       E02 =       4e+10  (2.225e+10, 5.775e+10)
       E03 =     4.5e+10  (3.752e+10, 5.248e+10)
       E04 =     3.8e+10  (3.38e+10, 4.22e+10)
       E05 =     2.6e+10  (2.432e+10, 2.768e+10)
       E06 =     2.4e+10  (2.255e+10, 2.545e+10)
       E07 =     1.9e+10  (1.794e+10, 2.006e+10)
```

```
        E08 =     1.6e+10  (1.519e+10, 1.681e+10)
        E09 =     7.6e+09  (7.401e+09, 7.799e+09)
        E10 =     6.1e+09  (5.909e+09, 6.291e+09)
        E11 =    2.85e+09  (2.763e+09, 2.937e+09)

gof =

            sse: 3.2946e-22
        rsquare: 0.9999
            dfe: 122
    adjrsquare: 0.9999
           rmse: 1.6433e-12
```




Prony Series Fit to Creep Data
x 10^-10
Test Data
Curve Fit
Creep Compliance (1/Pa)
Time (min)


Moduli Used in Prony Series Representation
x 10^10
Modulus (Pa)
Time Constant (min)

Published with MATLAB® R2013b

Appendix D

Inverse Laplace Transforms

```
%Inverse_Laplace.m
%By Eddie Parkman
%MSME Student, University of Texas at Arlington
%Dr. Ashfaq Adnan, Advising Professor

clear

%Input Function in Laplace Domain
syms s E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 tau1 tau2 tau3 tau4 tau5 tau6 tau7
tau8 tau9 tau10 tau11 C
%
Function1 = 1/(s/E0 + s/E1 + s^2/E1/(s+1/tau1))
Inverse1 = ilaplace(Function1)

Function2 = 1/(s/E0 + s/E1 + s/E2 + s^2/E1/(s+1/tau1) + s^2/E2/(s+1/tau2))
Inverse2 = ilaplace(Function2)

Function3 = 1/(s/E0 + s/E1 + s/E2 + s/E3 + s^2/E1/(s+1/tau1) + s^2/E2/(s+1/tau2) +
s^2/E3/(s+1/tau3))
Inverse3 = ilaplace(Function3)

% Function11 = 1/(s/E0 + s/E1 + s/E2 + s/E3 + s/E4 + s/E5 + s/E6 + s/E7 + s/E8 +
s/E9 + s/E10 + s/E11 + s^2/E1/(s+1/tau1) + s^2/E2/(s+1/tau2) + s^2/E3/(s+1/tau3) +
s^2/E4/(s+1/tau4) + s^2/E5/(s+1/tau5) + s^2/E6/(s+1/tau6) + s^2/E7/(s+1/tau7) +
s^2/E8/(s+1/tau8) + s^2/E9/(s+1/tau9) + s^2/E10/(s+1/tau10) + s^2/E11/(s+1/tau11))
% Inverse11 = ilaplace(Function11)
```

```
Function1 =

1/(s/E0 + s/E1 + s^2/(E1*(s + 1/tau1)))


Inverse1 =

(E0*E1)/(E0 + E1) - (E0^2*E1*tau1*exp(-(t*(E0 + E1))/(2*E0*tau1 + E1*tau1)))/
((2*E0*tau1 + E1*tau1)*(E0 + E1))


Function2 =

1/(s/E0 + s/E1 + s/E2 + s^2/(E1*(s + 1/tau1)) + s^2/(E2*(s + 1/tau2)))


Inverse2 =

(E0*E1*E2)/(E0*E1 + E0*E2 + E1*E2) - (exp(-(t*(E0*E1*tau1 + 2*E0*E1*tau2 +
2*E0*E2*tau1 + E0*E2*tau2 + E1*E2*tau1 + E1*E2*tau2))/(4*E0*E1*tau1*tau2 +
```

```
4*E0*E2*tau1*tau2 + 2*E1*E2*tau1*tau2))*(tau1*tau2*E0^2*E1^2*E2 +
tau1*tau2*E0^2*E1*E2^2)*(cosh((t*((E0^2*E1^2*tau1^2)/4 - E0^2*E1^2*tau1*tau2 +
E0^2*E1^2*tau2^2 + E0^2*E1*E2*tau1^2 - (3*E0^2*E1*E2*tau1*tau2)/2 +
E0^2*E1*E2*tau2^2 + E0^2*E2^2*tau1^2 - E0^2*E2^2*tau1*tau2 + (E0^2*E2^2*tau2^2)/4 +
(E0*E1^2*E2*tau1^2)/2 - (3*E0*E1^2*E2*tau1*tau2)/2 + E0*E1^2*E2*tau2^2 +
E0*E1*E2^2*tau1^2 - (3*E0*E1*E2^2*tau1*tau2)/2 + (E0*E1*E2^2*tau2^2)/2 +
(E1^2*E2^2*tau1^2)/4 - (E1^2*E2^2*tau1*tau2)/2 + (E1^2*E2^2*tau2^2)/4)^(1/2))/
(tau1*tau2*(2*E0*E1 + 2*E0*E2 + E1*E2))) - (tau1*tau2*sinh((t*
((E0^2*E1^2*tau1^2)/4 - E0^2*E1^2*tau1*tau2 + E0^2*E1^2*tau2^2 + E0^2*E1*E2*tau1^2 -
 (3*E0^2*E1*E2*tau1*tau2)/2 + E0^2*E1*E2*tau2^2 + E0^2*E2^2*tau1^2 -
E0^2*E2^2*tau1*tau2 + (E0^2*E2^2*tau2^2)/4 + (E0*E1^2*E2*tau1^2)/2 -
(3*E0*E1^2*E2*tau1*tau2)/2 + E0*E1^2*E2*tau2^2 + E0*E1*E2^2*tau1^2 -
(3*E0*E1*E2^2*tau1*tau2)/2 + (E0*E1*E2^2*tau2^2)/2 + (E1^2*E2^2*tau1^2)/4 -
(E1^2*E2^2*tau1*tau2)/2 + (E1^2*E2^2*tau2^2)/4)^(1/2))/(tau1*tau2*(2*E0*E1 +
2*E0*E2 + E1*E2)))*((E0*E1*tau1 + 2*E0*E1*tau2 + 2*E0*E2*tau1 + E0*E2*tau2 +
E1*E2*tau1 + E1*E2*tau2)/(4*E0*E1*tau1*tau2 + 4*E0*E2*tau1*tau2 +
2*E1*E2*tau1*tau2) - (tau2*E0^2*E1^2*E2 + tau1*E0^2*E1*E2^2)/
(tau1*tau2*E0^2*E1^2*E2 + tau1*tau2*E0^2*E1*E2^2))*(2*E0*E1 + 2*E0*E2 + E1*E2))/
((E0^2*E1^2*tau1^2)/4 - E0^2*E1^2*tau1*tau2 + E0^2*E1^2*tau2^2 + E0^2*E1*E2*tau1^2 -
 (3*E0^2*E1*E2*tau1*tau2)/2 + E0^2*E1*E2*tau2^2 + E0^2*E2^2*tau1^2 -
E0^2*E2^2*tau1*tau2 + (E0^2*E2^2*tau2^2)/4 + (E0*E1^2*E2*tau1^2)/2 -
(3*E0*E1^2*E2*tau1*tau2)/2 + E0*E1^2*E2*tau2^2 + E0*E1*E2^2*tau1^2 -
(3*E0*E1*E2^2*tau1*tau2)/2 + (E0*E1*E2^2*tau2^2)/2 + (E1^2*E2^2*tau1^2)/4 -
(E1^2*E2^2*tau1*tau2)/2 + (E1^2*E2^2*tau2^2)/4)^(1/2)))/((E0*E1 + E0*E2 + E1*E2)*
(2*E0*E1*tau1*tau2 + 2*E0*E2*tau1*tau2 + E1*E2*tau1*tau2))


Function3 =

1/(s/E0 + s/E1 + s/E2 + s/E3 + s^2/(E1*(s + 1/tau1)) + s^2/(E2*(s + 1/tau2)) + s^2/
(E3*(s + 1/tau3)))


Inverse3 =

(E0*E1*E2*E3)/(E0*E1*E2 + E0*E1*E3 + E0*E2*E3 + E1*E2*E3) -
(E0^2*E1*E2^2*E3^2*tau1*sum(exp(r3*t)/(E0*E1*E2*tau1 + E0*E1*E2*tau2 +
E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2 + 2*E0*E2*E3*tau1 + E0*E1*E3*tau3
+ E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 + E1*E2*E3*tau2 + E1*E2*E3*tau3 +
2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 + 4*E0*E1*E3*r3*tau1*tau2 +
4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 + 4*E0*E2*E3*r3*tau1*tau2 +
4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 + 2*E1*E2*E3*r3*tau1*tau2 +
2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 + 2*E1*E2*E3*r3*tau2*tau3 +
6*E0*E1*E2*r3^2*tau1*tau2*tau3 + 6*E0*E1*E3*r3^2*tau1*tau2*tau3 +
6*E0*E2*E3*r3^2*tau1*tau2*tau3 + 3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf
(2*E0*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E2*s3^3*tau1*tau2*tau3 + E1*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E2*E3*s3^2*tau1*tau3 + 2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 +
2*E0*E1*E2*s3^2*tau2*tau3 + 2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 +
E1*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau2 + E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 +
```

```
2*E0*E2*E3*s3*tau1 + 2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 +
E1*E2*E3*s3*tau2 + E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 +
E0*E1*E3*s3*tau3 + E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 +
E1*E2*E3 + E0*E2*E3 + E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2*E3^2*tau2*sum(exp
(r3*t)/(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 +
2*E0*E1*E3*tau2 + 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 +
E0*E2*E3*tau3 + E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 +
4*E0*E1*E2*r3*tau1*tau3 + 4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 +
2*E0*E1*E3*r3*tau1*tau3 + 4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 +
4*E0*E2*E3*r3*tau1*tau3 + 2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 +
2*E1*E2*E3*r3*tau1*tau3 + 2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3
+ 6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2^2*E3*tau3*sum(exp(r3*t)/(E0*E1*E2*tau1 +
E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2 + 2*E0*E2*E3*tau1
+ E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 + E1*E2*E3*tau2 +
E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1*E2^2*E3^2*tau1*tau2*sum((r3*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
```

```
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2*E3^2*tau1*tau2*sum((r3*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1*E2^2*E3^2*tau1*tau3*sum((r3*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2^2*E3*tau1*tau3*sum((r3*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
```

```
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2*E3^2*tau2*tau3*sum((r3*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2^2*E3*tau2*tau3*sum((r3*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
```

```
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1*E2^2*E3^2*tau1*tau2*tau3*sum((r3^2*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2*E3^2*tau1*tau2*tau3*sum((r3^2*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)) + E0^2*E1^2*E2^2*E3*tau1*tau2*tau3*sum((r3^2*exp(r3*t))/
(E0*E1*E2*tau1 + E0*E1*E2*tau2 + E0*E1*E3*tau1 + 2*E0*E1*E2*tau3 + 2*E0*E1*E3*tau2
+ 2*E0*E2*E3*tau1 + E0*E1*E3*tau3 + E0*E2*E3*tau2 + E1*E2*E3*tau1 + E0*E2*E3*tau3 +
E1*E2*E3*tau2 + E1*E2*E3*tau3 + 2*E0*E1*E2*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau1*tau3 +
4*E0*E1*E3*r3*tau1*tau2 + 4*E0*E1*E2*r3*tau2*tau3 + 2*E0*E1*E3*r3*tau1*tau3 +
4*E0*E2*E3*r3*tau1*tau2 + 4*E0*E1*E3*r3*tau2*tau3 + 4*E0*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau1*tau2 + 2*E0*E2*E3*r3*tau2*tau3 + 2*E1*E2*E3*r3*tau1*tau3 +
2*E1*E2*E3*r3*tau2*tau3 + 6*E0*E1*E2*r3^2*tau1*tau2*tau3 +
6*E0*E1*E3*r3^2*tau1*tau2*tau3 + 6*E0*E2*E3*r3^2*tau1*tau2*tau3 +
3*E1*E2*E3*r3^2*tau1*tau2*tau3), r3 in RootOf(2*E0*E2*E3*s3^3*tau1*tau2*tau3 +
2*E0*E1*E3*s3^3*tau1*tau2*tau3 + 2*E0*E1*E2*s3^3*tau1*tau2*tau3 +
```

```
E1*E2*E3*s3^3*tau1*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau3 +
2*E0*E1*E3*s3^2*tau2*tau3 + 2*E0*E2*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau2*tau3 +
2*E0*E1*E3*s3^2*tau1*tau2 + 2*E0*E1*E2*s3^2*tau1*tau3 + E1*E2*E3*s3^2*tau2*tau3 +
E1*E2*E3*s3^2*tau1*tau3 + E0*E2*E3*s3^2*tau2*tau3 + E1*E2*E3*s3^2*tau1*tau2 +
E0*E1*E3*s3^2*tau1*tau3 + E0*E1*E2*s3^2*tau1*tau2 + 2*E0*E2*E3*s3*tau1 +
2*E0*E1*E3*s3*tau2 + 2*E0*E1*E2*s3*tau3 + E1*E2*E3*s3*tau3 + E1*E2*E3*s3*tau2 +
E0*E2*E3*s3*tau3 + E1*E2*E3*s3*tau1 + E0*E2*E3*s3*tau2 + E0*E1*E3*s3*tau3 +
E0*E1*E3*s3*tau1 + E0*E1*E2*s3*tau2 + E0*E1*E2*s3*tau1 + E1*E2*E3 + E0*E2*E3 +
E0*E1*E3 + E0*E1*E2, s3)))/(E0*E1*E2 + E0*E1*E3 + E0*E2*E3 + E1*E2*E3)
```

Published with MATLAB® R2013b

References

[1] Saudi Basic Industries Corporation (SABIC), "ULTEM™ Resin," 2006-2015. [Online]. Available: http://www.sabic-ip.com/gep/Plastics/en/ProductsAndServices/ProductLine/ultem.html. [Accessed 22 March 2015].

[2] Saudi Basic Industries Corporation (SABIC), "SABIC Innovative Plastics - EDD," 2006-2015. [Online]. Available: http://www.sabic-ip.com/gepapp/eng/eddinter/edd. [Accessed 24 March 2015].

[3] A. C. Ugural and S. K. Fenster, Advanced Mechanics of Materials and Applied Elasticity, 5th ed., Upper Saddle River, NJ: Prentice Hall, 2012.

[4] H. F. Brinson and L. C. Brinson, Polymer Engineering Science and Viscoelasticity, New York, NY: Springer Science+Business Media, LLC, 2008.

[5] R. M. Christenson, Theory of Viscoelasticity, 2nd ed., Mineola, NY: Dover Publications, Inc., 2003.

[6] J. A. Foreman, C. J. Lundgren and P. S. Gill, "Measurement of the Physical Properties of Engineering Thermoplastics Using Thermal Analysis," TA Instruments.

[7] TA Instruments, "Thermal Analysis," 2010. [Online]. Available: http://www.tainstruments.com/pdf/brochure/dma.pdf. [Accessed 25 March 2015].

[8] S. P. Timoshenko and J. N. Goodier, Theory of Elasticity, 3rd ed., New York: McGraw-Hill Book Company, 1970.

[9] R. D. Bradshaw and L. C. Brinson, "A Sign Control Method for Fitting and Interconverting Material Functions for Linear Viscoelastic Solids," *Mech. Time Dep. Matls.,* vol. 1, pp. 85-108, 1997.

Biographical Information

Eddie Parkman earned his BSME from the University of Texas at Austin in 2004. Since then, he's worked at Corning Optical Communications LLC. At Corning, he's worked extensively on tight tolerance optical connectivity products, including injection molded, machined, die cast, and deep drawn components and assemblies. His professional background includes project management and leading product and process development teams for some of Corning's highest priority projects.

Eddie was licensed as a Professional Engineer in 2010. His technical interests include finite element modeling, composite materials & assemblies, and mechanics of materials topics such as viscoelasticity, plasticity, and viscoplasticity.