2020

Ensuring Positive Definiteness in Linear Viscoelastic Material Functions Based on Prony Series

Christopher D. Rehberg
University of Central Florida

ENSURING POSITIVE DEFINITENESS IN LINEAR VISCOELASTIC MATERIAL FUNCTIONS BASED ON PRONY SERIES

by

CHRISTOPHER DANIEL REHBERG

A thesis submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mechanical and Aerospace Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

ABSTRACT

This thesis presents a method to correct for non-positive-definiteness in linear viscoelastic material functions. Viscoelastic material functions for anisotropic materials need to be interconverted in a matrix coefficient prony series form, with a requirement of positive definiteness. Fitting is usually done as a uniaxial prony series, resulting in scalar coefficients. When these uniaxial coefficients are placed in a coefficient matrix, the required positive definiteness cannot be guaranteed. For those matrices that do not meet this requirement, finding the nearest symmetric semi-positive definite form of the matrix results in a viable prony series matrix coefficient with the required positive definiteness. These corrected prony series coefficients allow for material functions to be interconverted with minimal changes to experimental data.

ACKNOWLEDGMENTS

I would like to show my deepest gratitude to my thesis chair, Dr. Kawai Kwok. All the time, effort, and direction he has given to me in this thesis allowed me to grow as a student and complete this monumental task.

I am also indebted to my wife, who not only supported me during this thesis but encouraged me when I felt overwhelmed with the writing. Additionally, I must thank her for leading me to this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Viscoelastic materials have been growing in use, from polymers to polymer-based composites, to new shape-memory materials. The ever-increasing use of viscoelastic materials has heightened the demand for the ability to gather material properties to engineer and model these growing lists of materials. Specific techniques are required to define and model these materials accurately.

1.1 Elastic Analysis

In order to understand viscoelastic materials, an overview of elastic material analysis will first be undertaken. This will allow for an understanding of how viscoelastic material functions differ from elastic material functions. These functions can be found using a simple uniaxial tension test. This test uses a specimen designed to ensure a homogeneous state of stress and strain within the region to be measured. With a known length and cross-section area, a set force may be applied, and then stress, strain, and Young's modulus may be found. Other properties may also be found, but for this thesis, these properties are not being reviewed.

The first material quantity to be reviewed is that of engineering stress, which can be found by dividing applied tensile force, F, by the cross-sectional area, A_0 of our measured region.

$$\sigma_{av} = \frac{F}{A_0} \tag{1}$$

The second material quantity is engineering strain, which is determined by dividing the change of length, ΔL, of the measured region by the original length L_0 of the same region.

$$\varepsilon_{av} = \frac{L - L_0}{L_0} = \frac{L}{L_0} - 1 \tag{2}$$

When elastic materials vary stress linearly with the strain, Young's modulus, E, can be defined as the slope of the stress-strain curve. This modulus allows for a direct relationship between stress and strain, Hooke's law. An important note: with elastic materials, the quantities reviewed are not time-dependent.

$$E = \frac{\sigma_{av}}{\varepsilon_{av}} \tag{3}$$

$$\sigma_{av} = E\varepsilon_{av} \tag{4}$$

1.2 Viscoelastic Analysis

The most significant difference between elastic and viscoelastic materials is the time-dependent nature of the material functions. This requires different tests and analyses than used for elastic material functions. These tests are the relaxation and creep tests and are used to measure the time-dependent material functions.

For a relaxation test, a constant strain is applied quasi-statically to a uniaxial tensile bar. The bar is then stretched to a new fixed length, with the stain applied as close to instantaneous as possible

without inertial or dynamic effects. In this test, it is assumed that no previous stress or strain history exists in the test material. When the test material is loaded with this new strain, a stress response happens, but over time this stress response lessens until a constant value is reached.

.

$$E(t) = \frac{\sigma(t)}{\varepsilon_0} \tag{5}$$

$$\sigma(t) = \varepsilon_0 E(t) \tag{6}$$

Equation 5 is known as the relaxation modulus and describes the uniaxial stress-strain relationship. This is a viscoelastic analogous for Hooke's law, found in equation 4.

The creep test, for uniaxial materials, is found using the same procedure as the relaxation test. Rather than loading a strain, constant stress is loaded with an increasing strain response. After a long time, the strain response reaches a constant value. Equation (7) is the creep compliance found from the creep test.

$$D(t) = \frac{\varepsilon(t)}{\sigma_0} \tag{7}$$

$$\varepsilon(t) = \sigma_0 D(t) \tag{8}$$

1.3 Interconversion

The ever-increasing use of viscoelastic materials requires the ability to define creep and relaxation material functions. While these material functions can be found experimentally, the relaxation and creep tests can be expensive and time-consuming, usually requiring multiple runs with different temperatures to capture the full range of time in a reasonable amount of time. This constraint, together with either the creep or relaxation test being a challenge to find for the material function encourages the need to find new ways to determine creep or relaxation. For linear viscoelastic materials, the creep and relaxation material functions can be interconverted between each other, allowing for only one material function to be experimentally found.

The interconversion of material functions has one of its necessities in the possible inaccessibility of direct experimental results for one of these functions. An example of this can be found with the use of a constant-strain relaxation test. Finding the response of a stiff material subjected to a specified deformation can be problematic. (Park & Schapery, 1999); the same material could have the creep function easily measured with a constant-stress creep test. The creep function becomes the source function of an interconversion used to recreate the relaxation function as the target. This interconversion is governed by the equation below.

$$\int_0^t \boldsymbol{E}(t-\tau) \cdot \boldsymbol{D}(t) d\tau = t\boldsymbol{I} \tag{9}$$

With the increased use of interconversion methods for linear viscoelastic materials, a need for accurate approximations of the source and target functions has appeared. Presently the source function is found with experimental data, fit into a prony series, equation (10) for relaxation, and equation (11) for creep.

$$\boldsymbol{E}(t) = \boldsymbol{E}^{(0)} + \sum_{n=1}^{N} \boldsymbol{E}^{(n)} \exp(-t\rho_n) \tag{10}$$

$$\boldsymbol{D}(t) = \boldsymbol{D}^{(0)} + \sum_{m=1}^{M} \boldsymbol{D}^{(m)}[1 - \exp(-t\lambda_m)] \tag{11}$$

The prony series is finally used in an interconversion for the target function uniaxially. This approach is useful for isotropic materials but presents problems once anisotropic materials are to be interconverted.

When material functions, for anisotropic materials, are experimentally found, the coefficients for each material function curve must then be placed into unified matrix coefficients for an anisotropic prony series. This approach does not consider the need for prony series coefficients to be constrained positive definitely. This lack of constraint can keep useful interconversion algorithms from being utilized, such as those presented by Levesque(Jacques Luk-Cyr et al., 2013).

CHAPTER 2: LITERATURE REVIEW

(Lee & Knauss, 2000) demonstrate a method for manipulating, with a recursion formula, data gathered from an initial ramp. In laboratory environments, a ramp step, in stress or strain, is used to gather material properties, instead of a unit step. This ramp step becomes approximately the same as a unit step after a time interval around ten times the ramp-up time. While the approximation is useful, the initial loss of accuracy of the ramp-up time can result in the loss of a substantial portion of data. This recursion formula allows for the restoration of this lost data with a good result.

(Knauss & Zhao, 2007) show a simple method to increase the range of data that is acquired from a single test at a single temperature. Data acquired in laboratory experiments can be shorter than desired because of equipment demand or stability. With the ten-times rule, this leaves a shortened set of accurate data. With the addition of accelerating creep and relaxation, this leaves the shortened data set with only a couple of decades of data. An extended amount of time, five or more decades, is usually desired.

The use of computers and the commercially available Matlab code for the Trust Region Method, has allowed the accurate interpretation of data past the recorded decade. This method allows for the researcher's choice of time constants or to have the Trust Region Method determine a set of time constants. Further, it is shown that the use of only ten to fifteen data points and two time constants per decade are needed for the results.

Luk-Cyr, Crochon, Li, and Levesque (Jacques Luk-Cyr et al., 2013) present a set of algorithms for the interconversion of linear viscoelastic material functions of unidimensional and tridimensional materials. Four algorithms are developed, with a set of two for unidimensional materials and a second set for tridimensional materials. Each set has an interconversion from the creep to relaxation and from relaxation to creep. These algorithms depend on the equations for the thermodynamics of irreversible processes together with a prony series representation to achieve a method for interconversion of material functions with a highly accurate analytical result.

CHAPTER 3: PROBLEM DEFINITION

In today's aerospace environment, the increasing use of non-traditional materials are being used; among these materials are viscoelastic materials. Viscoelastic materials have shown an increased use in the aerospace industry, from carbon-fiber-reinforced polymers for plane shells and deployable space structures, to thin-film polymers for superpressure balloons. The increase in the dependence of viscoelastic materials results in an increased need to model.

The need to model viscoelastic materials requires either the creep compliance or the relaxation modulus of a specific material. For sensible reasons, many materials have the creep compliance measured from creep-recovery tests. This, however, presents a problem with the use of finite element modeling, as packages require knowing the relaxation modulus for the software's implementation. Fortunately, the viscoelastic material functions can be interconverted, allowing for the need to acquire only one function to have both.

3.1: Specific Problem

While present tests will work for isotropic materials, anisotropic materials present a different challenge. While the multidimensional properties can be obtained, when each dimension is fitted to a prony series with equal time constants, and placed in a coefficient matrix, shown below in a general form,

$$A = A' = \begin{bmatrix} a_{1,1} & \cdots & a_{1,j} \\ \vdots & \ddots & \vdots \\ a_{i,1} & \cdots & a_{i,j} \end{bmatrix} \tag{12}$$

an important matrix property, or positive definiteness, is not guaranteed. This matrix property is one found naturally in the coefficient matrix, arising with mechanically stability, if the stiffness matrix maintains positive definiteness. This property is also necessary for the interconversion algorithms, used in this thesis, to convert the anisotropic material functions.

3.2: Proposed Solution

The approach presented in this thesis is to correct the lack of positive definiteness obtained from laboratory tests when individual dimensions are fitted into a prony series and then placed into a coefficient matrix. This method allows the material properties to be measured and fitted using the methods considered best for the materials and tests, with the constraint that each dimension must use the same time constants. Next, each coefficient matrix is checked for positive definiteness and if the property is not found, to adjust the offending matrix to its nearest positive definiteness state.

CHAPTER 4: METHOD DEVELOPMENT

This chapter will focus on the development of the method, starting with identifying matrix properties and reviewing the assumption leading to this model. Next, the fundamental equation governing linear viscoelastic material function interconversion will be derived, this equation will not only allow for the interconversion, but becomes essential at the end of the model as a check that the data provided to the model was useful, and an accurate conversion was performed. After the fundamental equation, a look at the solution provided for the correction of matrix properties will be reviewed. A review of the algorithms used, in this model, to interconvert the material functions will be examined. Finally, an overview of the entire model, from start to finish, will be conducted.

4.1 Matrix Properties

At present, many materials have their material functions experimentally found as a uniaxial prony series. When these uniaxial series are placed in a multidimensional matrix form, no guarantee of positive definiteness is provided. A new method is required for the fitting of multidimensional material functions, in a prony series representation, in order to have these required properties of symmetrical positive definiteness.

The goal of this model is to allow the interconversion of material functions, after correcting problems in matrices that may arise from errors, such as noise from experimentation and

computational errors. This new model uses a numerical method approach for multidimensional representation to ensure the required properties, accomplish interconversion, and verify the results. For this thesis, MATLAB and Python have been utilized to ensure the method works, but to also allow for automation of the process.

4.2 Assumptions

The matrix coefficients of the prony series must be positive definite. When unidimensional fittings are placed in a tensor, this condition is not always met, because of experimental and numerical errors. This thesis assumes that those negative eigenvalues should have been very close to zero and positive. This, in turn, has led to the model used in this thesis, where coefficients that do not meet requirements are then adjusted to become positive definite with the smallest changes possible. Thankfully this process has been devised before for other problems.

4.3 Fundamental Equation of Linear Viscoelastic Interconversion

A key component of viscoelastic materials is the stress/strain relationship. Unlike elastic materials, the stress/strain relationship of viscoelastic materials varies with time. This leads to two material functions: a stress response called the relaxation modulus, and a strain response called the creep compliance. When a viscoelastic material is placed under a constant strain, the stress response will

decrease with time; this is the relaxation modulus. When a constant stress is applied to a viscoelastic material an increasing strain response is attained, the creep compliance. Equations (*13*) and (*14*) show the material functions.

Relaxation Modulus: $$\mathrm{E(t)} = \frac{\sigma(\mathrm{t})}{\varepsilon_0} \tag{13}$$

$Creep\ Compliance$: $$D(t) = \frac{\varepsilon(t)}{\sigma_0} \tag{14}$$

For linear viscoelastic materials, the creep and relaxation responses can be separated, and functions can be shown to be connected, allowing for the interconversion of the material functions. The start of these connections can be shown when the material function is rewritten to represent stress and strain, respectively:

$$\sigma(t) = \int_0^t E(t-\tau) \cdot \frac{d\varepsilon(\tau)}{d\tau} d\tau \tag{15}$$

$$\varepsilon(t) = \int_0^t D(t-\tau) \cdot \frac{d\sigma(\tau)}{d\tau} d\tau \tag{16}$$

Then by taking the Laplace transform of equation (*15*) and (*16*), the material functions become:

$$\bar{\varepsilon}(s) = S\bar{D}(s)\bar{\sigma}(s) \tag{17}$$

$$\bar{\sigma}(s) = S\bar{E}(s)\bar{\varepsilon}(s) \tag{18}$$

Substituting equation (*17*) into equation (*18*),

$$\bar{E}(s)\bar{D}(s) = \frac{1}{s^2} \tag{19}$$

By taking the inverse Laplace transformation of equation (*19*), we arrive at a Volterra equation of the first kind and the fundamental equation for the interconversion of linear viscoelastic materials

$$\int_0^t E(t)D(t-\tau)d\tau = t \tag{20}$$

$$\int_0^t E(t-\tau)D(t)d\tau = t \tag{21}$$

The relationships are shown in equation *(20)* or (*21*) are used in the building of not only the algorithms for the interconversion of material functions, but are also used as the basis of checking the results of the model. When converting the material functions as matrices, the previous governing equation is represented as the following, with t**I** being the t-multiplied identity matrix.

$$\int_0^t \boldsymbol{E}(t-\tau)\cdot\boldsymbol{D}(t)d\tau = t\boldsymbol{I} \tag{22}$$

Generally, and in this model, the creep compliance and relaxation modulus are represented as prony series, shown in equation (*23*) and (*24*). With $\boldsymbol{E}^{(0)}\ and\ \boldsymbol{D}^{(0)}$ being the equilibrium relaxation and instantaneous creep moduli respectively and $\rho_n\ and\ \lambda_m$ the inverted time constants.

$$\boldsymbol{E}(t) = \boldsymbol{E}^{(0)} + \sum_{n=1}^{N}\boldsymbol{E}^{(n)}\exp(-t\rho_n) \tag{23}$$

$$\boldsymbol{D}(t) = \boldsymbol{D}^{(0)} + \sum_{m=1}^{M}\boldsymbol{D}^{(m)}[1-\exp(-t\lambda_m)] \tag{24}$$

4.4 Eigenvalues

With all experimental data fitted as uniaxial prony series for the source function, and each dimension has the same time constants, the next step of the model can be performed. All uniaxial coefficients must be placed in a symmetric multidimensional prony series coefficient. This new matrix form follows the same equation as seen in equations (*23*) or (*24*), but does not have a guarantee of positive definiteness required. Each coefficient matrix must be checked for positive definiteness, by calculation of eigenvalues. Each matrix that has any negative eigenvalues then needs to be passed to the next step of this model, while those that do meet the requirements are left alone.

4.5 Nearest Symmetric Positive Semidefinite Matrix

Those matrices found not to be positive definite need to have this corrected. In this method, this is accomplished by finding the nearest symmetric positive semidefinite form of the matrix. This change is accomplished by an algorithm described by (Higham 1988).

This algorithm computationally finds the nearest symmetric positive semidefinite matrix to a real matrix using a modification of Halmos' formula. These changes are using a bisection method, applied to the formula, to compute an upper and lower bound of the distance from the symmetric positive matrix to the real matrix. From there, the formula is formulated as a zero-finding problem,

and a hybrid Newton-bisection algorithm is applied. The positive semidefinite property of the formulated matrix is checked with a Cholesky decomposition.

Once this process has been accomplished, our corrected source function can then be passed to the proper interconversion algorithm described by (J. Luk-Cyr et al., 2013) and shown in the next section.

4.6 Interconversion of Viscoelastic Material Functions

Using equation (*22*), to interconvert the source function to the target function can be a problem; this equation is a Volterra equation and is generally ill-posed, causing a problem for numerical integration. The numerical solution can be convergent, though it does not always converge to the proper solution (Sorvari & Malinen, 2007). This problem of ill-posed can be corrected by representing our functions as prony series. This turns equation (*22*) into a well-posed problem and allows for the solving of the target function coefficients.

As viscoelastic material functions are often obtained from creep-recovery tests, and finite element software usually requires knowing the relaxation function, this thesis will display the algorithm for converting creep to relaxation first (J. Luk-Cyr et al., 2013). The only change that can be found in the implementation shown below is for more than the tridimensional cases presented by Levesque and authors. This change is accomplished by replacing the hard coding of dimensions of a six by six matrix to that of variable size. This variable size is represented by R and is the N or M dimension size of the square matrices for the prony series coefficients.

4.6.1 Interconversion from D(t) to E(t)

The algorithm starts by computing the internal matrices for the creep compliance. Here, M is the number of coefficient matrices in the data, $c_{\mathcal{L}}$ is the Cholesky decomposition, and λ_m are the creep inverted time constants:

1: $\boldsymbol{A}^{(1)}_{[RxR]} = \boldsymbol{D}^{(0)}$

2: $\boldsymbol{A}^{(2)}_{[RxR\cdot M]} = \left[c_{\mathcal{L}}\left(\lambda_1 \boldsymbol{D}^{(1)}\right)\middle| c_{\mathcal{L}}\left(\lambda_2 \boldsymbol{D}^{(2)}\right)\middle| \ldots \middle| c_{\mathcal{L}}\left(\lambda_M \boldsymbol{D}^{(M)}\right)\right]$

3: $\boldsymbol{A}^{(3)}_{[R\cdot MxR\cdot M]} = \begin{bmatrix} [\lambda_1]_R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & [\lambda_M]_R \end{bmatrix}$

4: $\boldsymbol{B}_{[R\cdot MxR\cdot M]}$ *is an identity matrix*

The second stage of the algorithm then computes the internal matrices of the relaxation modulus.

5: $\boldsymbol{L}^{(1)} = \left(\boldsymbol{A}^{(1)}\right)^{-1}$

6: $\left(\boldsymbol{L}^{(2)}\right)^T = \left(\boldsymbol{A}^{(2)}\right)^T \cdot \left(\boldsymbol{A}^{(1)}\right)^{-1}$

7: $\boldsymbol{L}^{(3)} = \boldsymbol{A}^{(3)} + \left(\boldsymbol{A}^{(2)}\right)^T \cdot \left(\boldsymbol{A}^{(1)}\right)^{-1} \cdot \boldsymbol{A}^{(2)}$

Next, we calculate the eigenvectors **P** of $\boldsymbol{L}^{(3)}$ with singular value decomposition and use this result to computer two more matrices, we need to find the relaxation coefficients and inverted time constants.

8: $\boldsymbol{L}^{(3*)} = \boldsymbol{P}^T \cdot \boldsymbol{L}^{(3)} \cdot \boldsymbol{P}$

9: $\left(\boldsymbol{L}^{(2*)}\right)^T = \boldsymbol{P}^T \cdot \left(\boldsymbol{L}^{(2)}\right)^T$

Now we can obtain $\boldsymbol{C}^{(0)}, \boldsymbol{C}^{(n)}, and\ \rho_n$.

10: $\boldsymbol{E}^{(0)} = \boldsymbol{L}^{(1)} - \sum_{n=1}^{N} \boldsymbol{E}^{(n)}$

11: $\boldsymbol{for}\ n = 1\ to\ N = R \cdot M\ \boldsymbol{do}$

12: $E_{ij}^{(n)} = \frac{L_{in}^{(2*)} L_{jn}^{(2*)}}{L_{nn}^{(3*)}}$

13: $\rho_n = \frac{L_{nn}^{(3*)}}{B_{nn}}$

14: $\boldsymbol{end\ for}$

4.6.2 Interconversion from E(t) to D(t)

For the relaxation to creep algorithm, the same notations as before will be followed, including the Cholesky decomposition. The only change is that M takes the place of N and the use of our relaxation inverted time constants, ρ_n. Since the changes are nominal, all the steps will be presented without comments.

1: $\boldsymbol{L}_{[RxR]}^{(1)} = \boldsymbol{E}^{(0)} + \sum_{n=1}^{N} \boldsymbol{E}^{(n)}$

2: $L_{[RxR \cdot N]}^{(2)} = \left[c_{\mathcal{L}}\left(\rho_1 \boldsymbol{E}^{(1)}\right) \middle| c_{\mathcal{L}}\left(\rho_2 \boldsymbol{E}^{(2)}\right) \middle| \ldots \middle| c_{\mathcal{L}}\left(\rho_N \boldsymbol{E}^{(N)}\right)\right]$

3: $\boldsymbol{L}^{(3)}_{[R\cdot NxR\cdot N]} = \begin{bmatrix} [\rho_1]_R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & [\rho_N]_R \end{bmatrix}$

4: $\boldsymbol{B}_{[R\cdot MxR\cdot M]}$ *is an identity matrix*

5: $\boldsymbol{A}^{(1)} = \left(\boldsymbol{L}^{(1)}\right)^{-1}$

6: $\left(\boldsymbol{A}^{(2)}\right)^T = \left(\boldsymbol{L}^{(2)}\right)^T \cdot \left(\boldsymbol{L}^{(1)}\right)^{-1}$

7: $\boldsymbol{A}^{(3)} = \boldsymbol{L}^{(3)} + \left(\boldsymbol{L}^{(2)}\right)^T \cdot \left(\boldsymbol{L}^{(1)}\right)^{-1} \cdot \boldsymbol{L}^{(2)}$

8: *Compute the eigenvectors* $\boldsymbol{P}$ *of* $\boldsymbol{A}^{(3)}$ *with singular value decomposition*

9: $\boldsymbol{A}^{(3*)} = \boldsymbol{P}^T \cdot \boldsymbol{A}^{(3)} \cdot \boldsymbol{P}$

10: $\left(\boldsymbol{A}^{(2*)}\right)^T = \boldsymbol{P}^T \cdot \left(\boldsymbol{A}^{(2)}\right)^T$

11: $\boldsymbol{D}^{(0)} = \boldsymbol{A}^{(1)}$

12: $\boldsymbol{for}\ m = 1\ to\ M = R \cdot N\ \boldsymbol{do}$

13: $D_{ij}^{(m)} = \dfrac{A_{im}^{(2*)} A_{jm}^{(2*)}}{A_{mm}^{(3*)}}$

14: $\lambda_m = \dfrac{A_{mm}^{(3*)}}{B_{mm}}$

15: $\boldsymbol{end\ for}$

4.7 Convolution Check

Once the material source function has been interconverted to the desired target function, a check for validity of the interconversion needs to be performed. In this model, this check is carried out using the convolution integral (*20*) in the time domain. This check ensures that only proper data was converted. When incorrect data or assumptions are used, the correction algorithm will create a symmetric positive semidefinite matrix. This matrix can be converted by the algorithms designed by Levesque and team. The matrix produced by this interconversion will not provide values close to those expected from the convolution integral. This integration also provides the range of acceptable values that the interconversion algorithm produced.

4.8 Model Steps

The steps of this model to allow for the interconversion of material functions with the required properties are as follows: This process will start with the collection of experimental data for the source function. This multidimensional material data can then be fitted as a uniaxial prony series, with any method desired as long as the time constants are maintained in each dimension. After the fitting, the uniaxial prony series are then placed in symmetric matrix coefficients. The coefficient matrices are then checked for positive definiteness; those that do not meet this requirement are then fed to an algorithm to find the nearest semi-positive definiteness matrix. Now that all matrices

have the required properties, the proper interconversion algorithm can be applied. Finally, a check of source and target functions is conducted with the convolution integral.

CHAPTER 5: RESULTS

The data employed in the results section is from a linear low-density polyethylene employed in high altitude balloons found in the paper, Thermoviscoelastic Models for Polyethylene Thin Films (Li et al., 2016). This paper found the creep compliances were found utilizing three different tests. Uniaxial tension creep tests determined the linear in-plane creep compliance master curves. The uniaxial tension test at a constant strain rate was used to simulate the nonlinear behavior at larger strains. Finally, a biaxial bubble test was employed to characterize the behavior under stress conditions that more represent those found for a balloon in operational conditions. The transformed nominal stress and strains from these experimental tests were utilized to calculate the master creep compliance curves.

This paper presented twenty prony series coefficients for the creep compliance curves of D11, D13, D21, D22, D23, and D66. Initially, this thesis utilized these coefficients for a three by three matrix, but without the D33 compliance curve, the data was unable to be appropriately corrected. Therefore, the coefficients for the compliance curves for D11, D21, and D22 were placed in two by two matrix coefficients for an anisotropic prony series. This anisotropic creep matrix became the source function to be checked, corrected and interconverted.

5.1 Eigenvalues

With the source function placed in a prony series with matrix coefficients, the next step in the model can proceed, the checking of eigenvalues. Table 1 shows all the eigenvalues for the fitted data, with three of the matrices not having the required positive definiteness, with each matrix having one negative eigenvalue.

Table 1: Eigenvalues for Fitted Data

Coefficient Matrix	Eigenvalue 1	Eigenvalue 2
D^0	0.00045	0.00015
D^1	1.74857485e-04	1.09392515e-04
$\boldsymbol{D^2}$	**-5.23719140e-06**	**7.03970914e-05**
$\boldsymbol{D^3}$	**9.86870394e-05**	**-1.17010394e-05**
D^4	1.41011072e-04	8.74692808e-06
D^5	5.76576346e-05	1.51549365e-04
$\boldsymbol{D^6}$	**2.60431459e-04**	**-9.25845891e-06**
D^7	4.67931334e-04	7.62586665e-05
D^8	6.94166790e-04	2.08773210e-04
D^9	9.62506308e-04	1.62973692e-04
D^{10}	1.28266095e-03	1.78289055e-04
D^{11}	1.58297224e-03	1.85547765e-04
D^{12}	5.34614336e-04	1.82288566e-03

D^{13}	5.90493210e-04	1.75980679e-03
D^{14}	1.40788479e-03	9.86452121e-05
D^{15}	1.56928357e-03	3.14616428e-04
D^{16}	1.66932398e-03	4.23766021e-04
D^{17}	5.15584982e-04	2.64101502e-03
D^{18}	1.38879234e-03	3.17217656e-04
D^{19}	1.88468891e-04	4.59991093e-05

These three matrices were then run through the algorithm to find the nearest symmetric positive semidefinite version of the matrix, this results in the change of the negative eigenvalues and making each matrix conform to all properties that are required. Table 2 shows in bold those eigenvalues that have been changed by the model truncated to the fourth decimal.

Table 2: Fitted and Corrected Eigenvalues

Coefficient Matrix	Fitted Eigenvalues		Corrected Eigenvalues	
D^{2}	**-5.2371e-06**	7.0397e-05	**1.3552e-20**	7.0397e-05
D^{3}	9.8687e-05	**-1.1701e-05**	9.8687e-05	**2.0328e-20**
D^{6}	2.6043e-04	**-9.2584e-06**	2.6043e-04	**1.3552e-20**

Table 3 shows the original matrices and the corrected forms for the three coefficient matrices that were corrected in the model. While each location was changed, the shift in each dimension was small. This small shift, along with the minimal change in eigenvalues, reinforces the notation that errors cause the lack of positive definiteness, and not another source.

Table 3: Fitted and Corrected Matrices

D^2 Original	D^2 Corrected
$\begin{bmatrix} 6.5109e-06 & -2.7396e-05 \\ -2.7396e-05 & 5.8649e-05 \end{bmatrix}$	$\begin{bmatrix} 1.0934e-05 & -2.5499e-05 \\ -2.5499e-05 & 5.9462e-05 \end{bmatrix}$
D^3 Original	D^3 Corrected
$\begin{bmatrix} 6.2843e-05 & -5.1691e-05 \\ -5.1691e-05 & 2.4143e-05 \end{bmatrix}$	$\begin{bmatrix} 6.6642e-5 & -4.6211e-05 \\ -4.6211e-05 & 3.2044e-05 \end{bmatrix}$
D^6 Original	D^6 Corrected
$\begin{bmatrix} 1.5508e-04 & -1.3158e-04 \\ -1.3158e-04 & 9.6093e-05 \end{bmatrix}$	$\begin{bmatrix} 1.587e-04 & -1.2706e-04 \\ -1.2706e-04 & 10.173e-5 \end{bmatrix}$

5.2 Fitted data vs. Corrected Data

After correcting coefficient matrices, a check of the new creep compliance modulus over time can be done. This new correct compliance modulus can be compared to the compliance modulus produced from the experimental data. When this comparison is made, a small variance, in all dimensions, from the experimental data is found. This variance increases with time, becoming the greatest difference at the most extreme times. These variances are shown in Figure 1: Comparative

Creep Modulus over Time D11, Figure 2: Comparative Creep Modulus over Time D21, and Figure 3: Comparative Creep Modulus over Time D, for the D11, D21, and D22 dimensions.

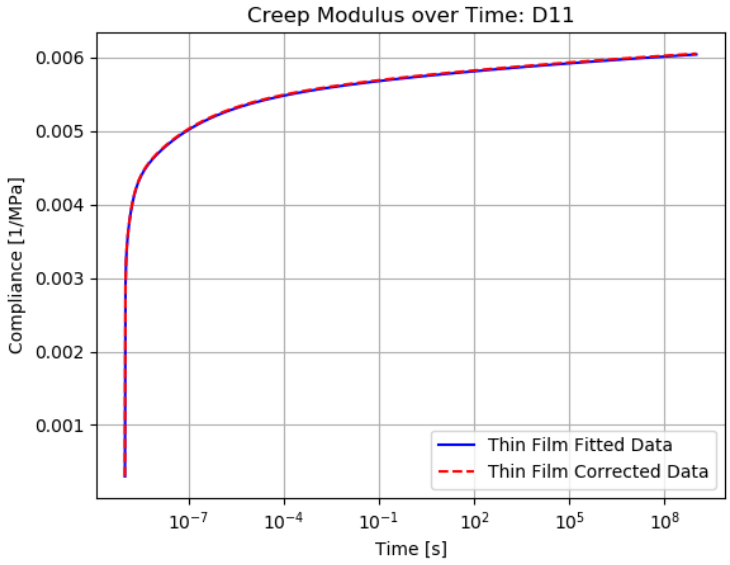

Figure 1: Comparative Creep Modulus over Time D11

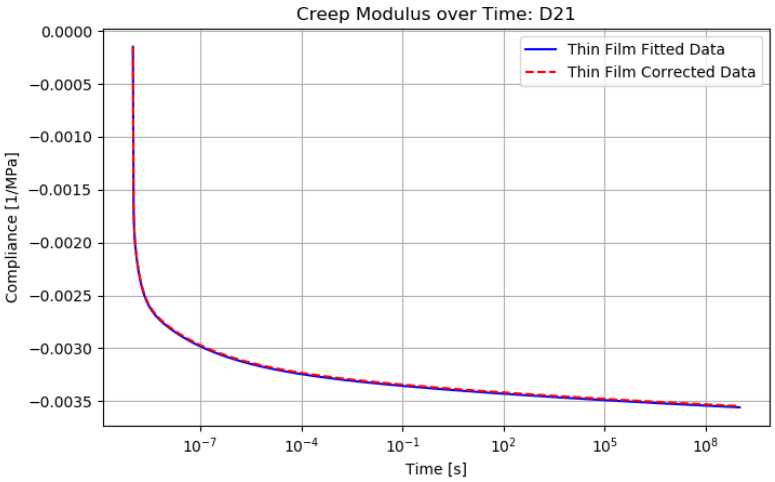

Figure 2: Comparative Creep Modulus over Time D21

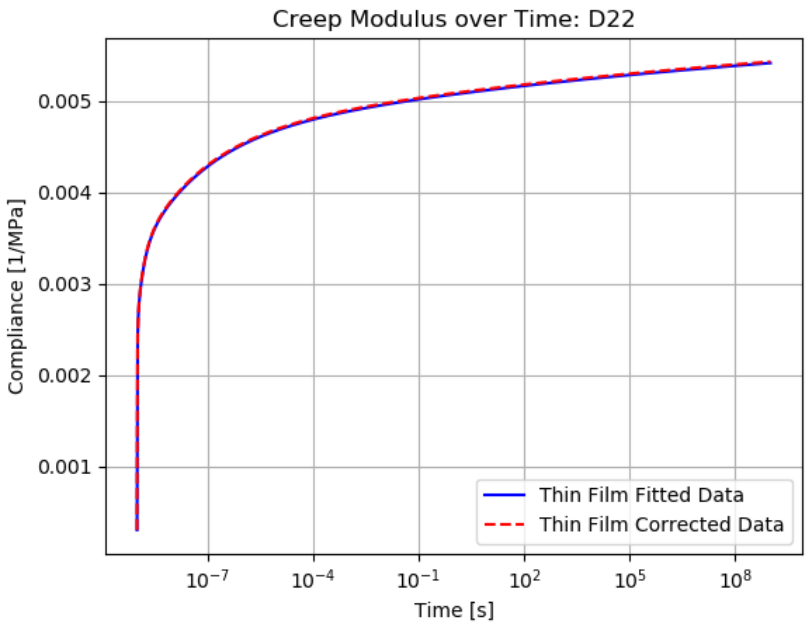

Figure 3: Comparative Creep Modulus over Time D22

5.2.1 R-squared

When looking for accuracy of the fit for the corrected compliance modulus compared to the experimental modulus, we can check the R squared values. With the D11 dimension, there is an R squared value of 0.9994. This is the best fit for all three dimensions. The shear D21 dimension has the worse fit with an R squared of 0.9982. Finally, the D22 dimension R squared was found to be 0.9992. These R squared show an extremely good fit for the corrected data when compared to the

experimental data. This helps to show that the model maintains accuracy while adding the required matrix properties.

5.2.2 Absolute Error Over Time

A final check of the accuracy of the corrected creep compliance modulus can be seen with graphing of absolute error over time for each dimension. Each of these graphs shows a small absolute error over the time range, with a consistent error, and a small amount of overestimating for each dimension.

We can also look at the global view of the absolute error, with the absolute percent error. For the D11 dimension, the percent error was found to be 0.2135%, with the D21 and D22 to be 0.3627% and 0.2954%, respectively. These minimal percent errors continue to show the model creates only minimal changes to moduli. This shows that the method developed in this thesis is a viable means to correct prony series coefficient matrices.

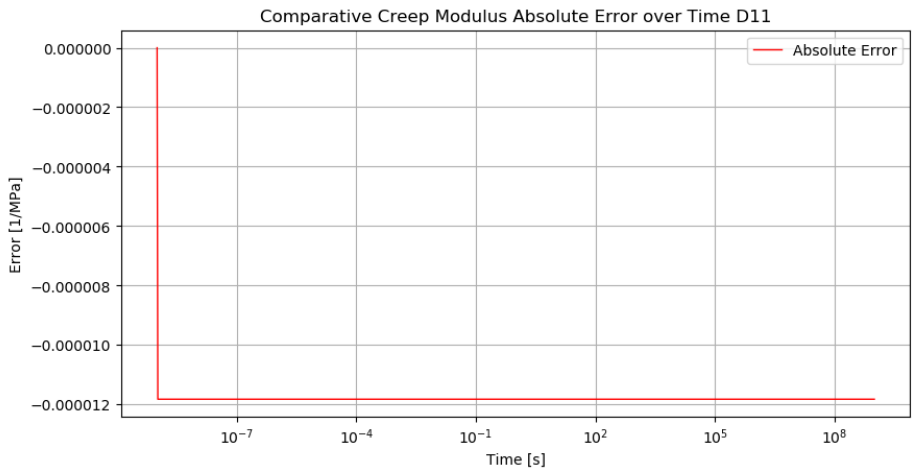

Figure 4: Comparative Creep Modulus Absolute Error over Time D11

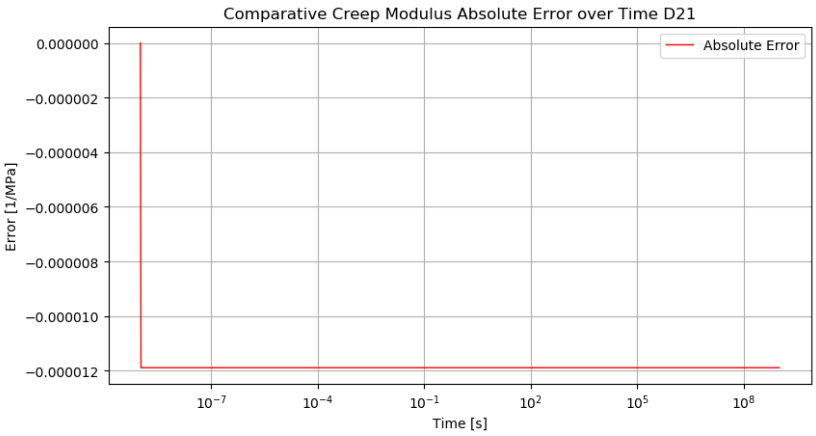

Figure 5: Comparative Creep Modulus Absolute Error over Time D21

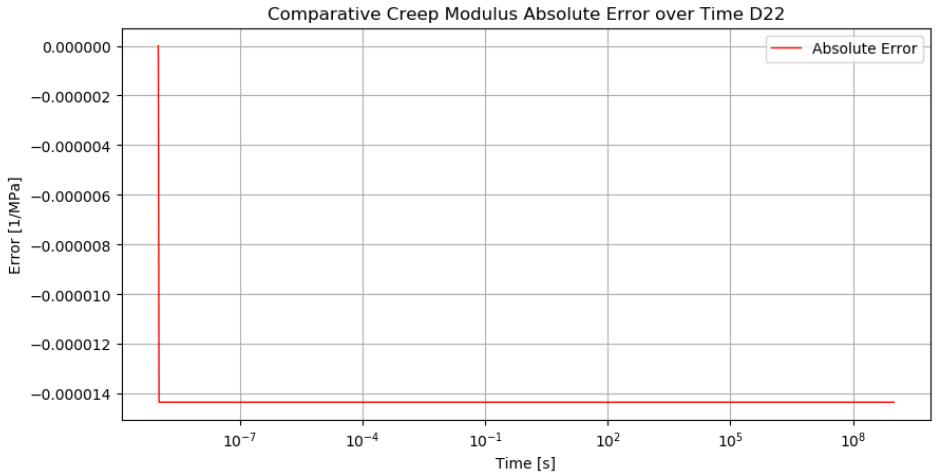

Figure 6: Comparative Creep Modulus Absolute Error over Time D22

5.3 Convolution

With the corrected data being shown to produce an acute representation of the modulus, the rest of the model can be followed. After correcting the matrices, the source function of creep compliance is converted to the relaxation function using the Levesque algorithm shown in section 4.6.1. As Levesque had already shown this algorithm to be accurate, this thesis does not show an account of accuracy. Instead, a check of the final conversion using the convolution integral and the errors found from the convolution are shown as the final check of the model.

Since we know, the convolution integral should have a solution of t**I**, which is the identity matrix multiplied by the scalar time applied to the convolution integral. This shows that over time, the in-

plane solutions for D11 and D22 should be shown to be the time given to the integral, while the shear D21 should be zero at all times. The solution of the convolution integral for each dimension from the model was then plotted against time to demonstrate if there was a departure of the answer at any time. All three dimensions were found to be close to the answer, with the extreme time range showing to have the furthest departure. This departure showed an increasing amount, from the expected value, at around 10e6 seconds.

This departure is to be expected, as the corrected creep compliance was shown to be furthest from the experimental data at the greater time range. This result was also expected numerically, as the implementation of the convolution integral was found to be more unstable and less accurate the greater the time variable.

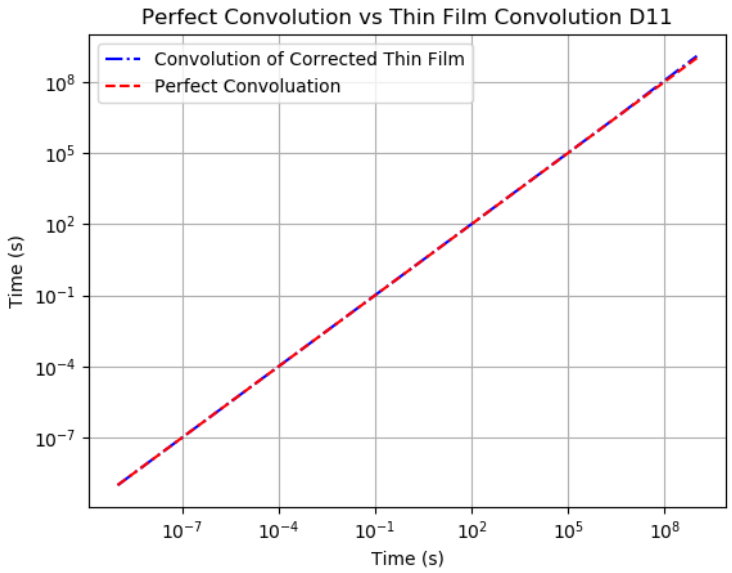

Figure 7: Perfect Convolution vs. Thin Film Convolution D11

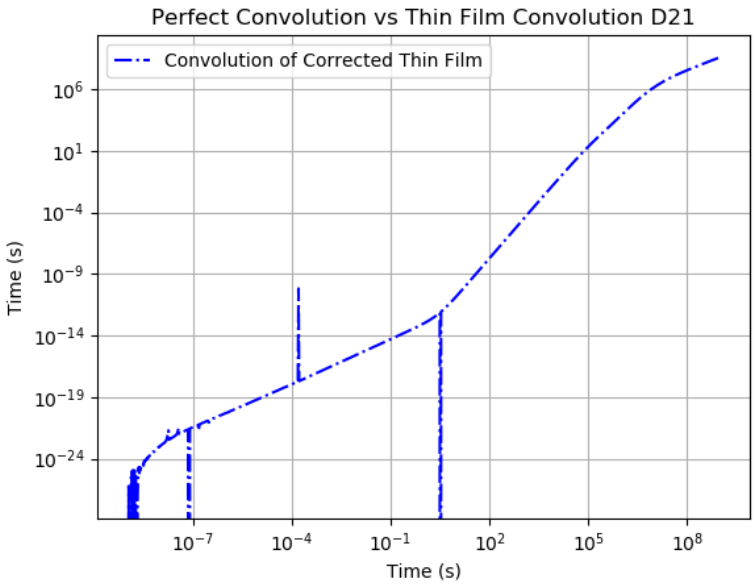

Figure 8: Perfect Convolution vs. Thin Film Convolution D21

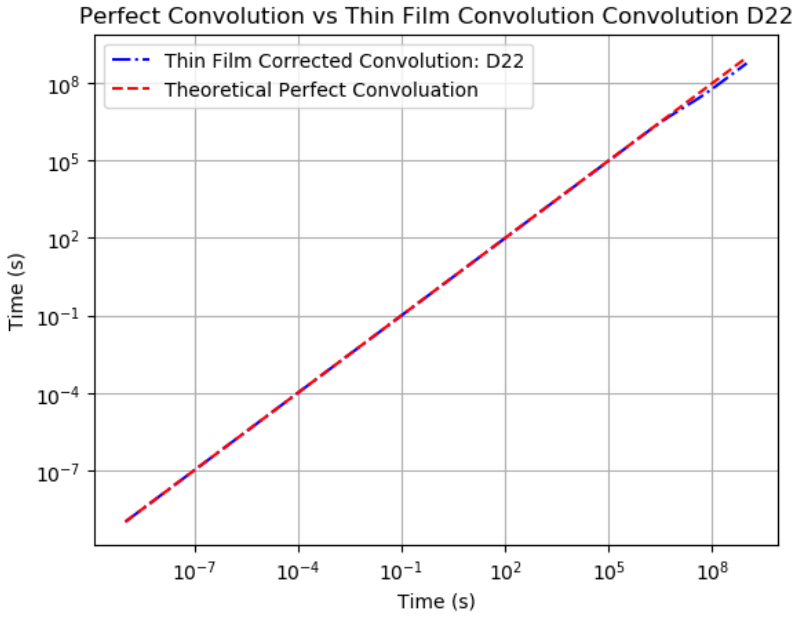

Figure 9: Perfect Convolution vs. Thin Film Convolution D22

5.3.1 Relative Error of Convolution

A final check of the convolution can be done with the relative error over time. This check is the normalization of the absolute error of the convolution normalized by the time step. These graphs show a negligible error until around 10e6 seconds, at which point each graph gains an error of no greater than the absolute value of 0.45, with the D22 coefficient showing a reduction in relative error at the end of the graph. This increase in error is from two separate sources. First, the corrected creep compliance modulus started to show the most significant difference at the same time frame as the relative error increases. Second, the numerical methods to check the convolution integral accumulates errors with larger time inputs. This is supported by each dimension showing the same shape and location of the accelerating error.

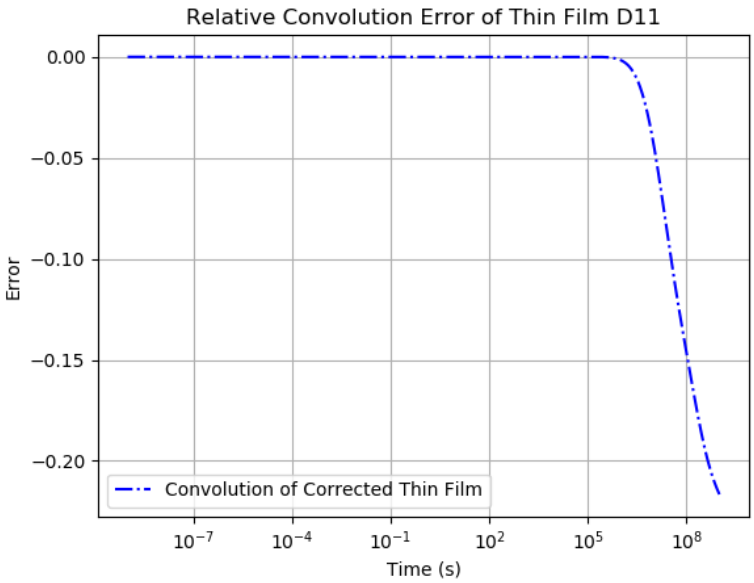

Figure 10: Relative Convolution Error of Thin Film D11

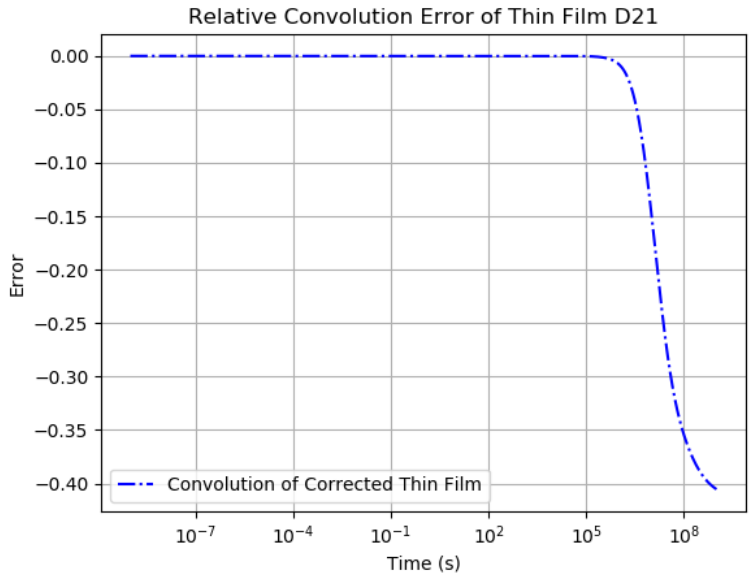

Figure 11: Relative Convolution Error of Thin Film D21

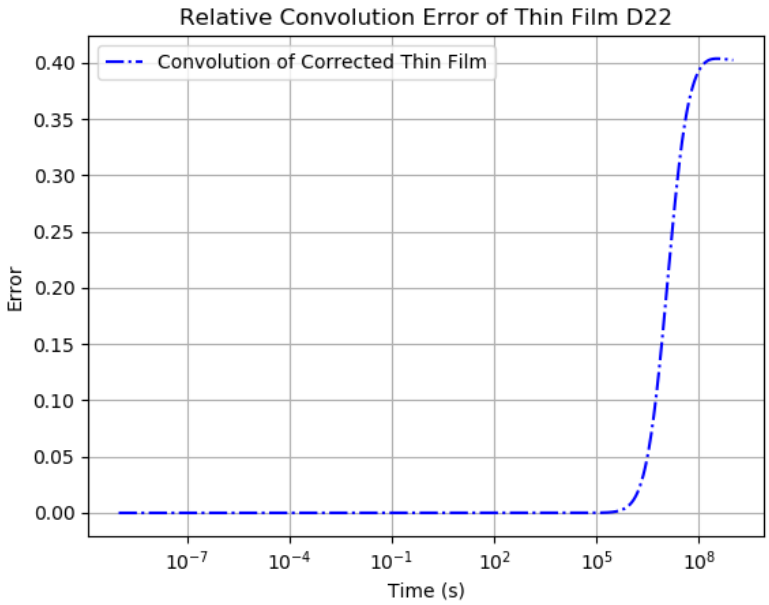

Figure 12: Relative Convolution Error of Thin Film D22

CHAPTER 6: CONCLUSION

This thesis has shown how uniaxially fitted prony series coefficients can be fitted into a matrix form to allow for the interconversion of linear viscoelastic material functions. The model presented in this paper corrects matrix coefficient eigenvalues, correcting the lack of positive definiteness, regardless of the method used to fit experimental data, if all time constants are maintained across dimensions. With minimal changes to experimental data, the presented model allows for interconversion of material functions while presenting minimal errors. This model also allows for either material function to serve as the source function for interconversion.

APPENDIX

APPENDIX A: PYTHON CODE

```
"""
Name: anisotropic-interconversion
Description: Python library to interconvert anisotropic relaxation and creep
prony series
Author: Christopher Rehberg
Email: christopher.rehberg@knights.ucf.edu
"""

# External libaries to run the viscoelastic interconversion library
import numpy as np
import pandas as pd
import scipy.integrate as integrate
import scipy.linalg as linalg
from numpy import linalg as la

def import_properties_excel(excel_file, coeff_size=None, invert=False):
    """Import matrices and creep/relaxation times from an excel
       format with a specific format for this library

    Parameters
    ----------
    excel_file : xlsx
        Excel file
    coeff_size : None or int
        sets the size of the matrix to be read, default to none
    invert : Bool
        flag if time consts need to be inverted

    Returns
    -------
    mat0 : numpy.array
        Instantaneous modulus
    matrix_coeff : numpy.array
        Modulus coefficients
    time_consts : numpy.array
        inverted time constants
    coeff_size : int
        Number of coefficent matrices
    """

    # Open the excel file with viscoelastic material properties
    with open(excel_file, 'br') as excel_loc:

        # Read in the number of matrix coefficients, and set as an integer
        num_coeff = pd.read_excel(excel_loc, header=None, usecols=[1], nrows=
1)
        num_coeff = int(num_coeff.values)

        # Check to see if a matrix size has been set, default to none
```

```
        if coeff_size is None:
            # Read in the size of the matrix coefficients, and set as an int
            coeff_size = pd.read_excel(
                excel_loc, header=None, usecols=[3], nrows=1)
            coeff_size = int(coeff_size.values)

        # Read in the instantious coefficient as a size of 6x6
        mat0 = pd.read_excel(excel_loc, header=None,
                             usecols=[0, 1, 2, 3, 4, 5], nrows=6, skiprows=5)
        # Change from dataframe to a numpy array
        mat0 = mat0.to_numpy()
        # Select only the required coefficients
        mat0 = mat0[:coeff_size, 0:coeff_size]

        # Create an empy array to store coefficients
        matrix_coeff = np.empty((num_coeff, coeff_size, coeff_size))

        for i in range(num_coeff):

            # Read a coefficient matrix from excel
            temp_coeff = pd.read_excel(excel_loc, header=None,
                                       usecols=[j+(6*i) for j in range(6)],
                                       nrows=6, skiprows=13)

            # Convert to a numpy array and select required coefficients
            temp_coeff = temp_coeff.to_numpy()

            matrix_coeff[i, :, :] = temp_coeff[0:coeff_size, 0:coeff_size]

        # Read in the time constants
        time_consts = pd.read_excel(excel_loc, header=None,
                                    usecols=[1+i for i in range(num_coeff)],
                                    nrows=1, skiprows=2)

        # Convert to a numpy array and reshape to a 1D array
        time_consts = time_consts.to_numpy()
        time_consts = np.reshape(time_consts, num_coeff)

        # Invert time consts if required
        if(invert):
            time_consts = 1 / time_consts

    return mat0, matrix_coeff, time_consts, coeff_size

def StoC(S0, S_mats, lambdas, coeff_size):
    """Converts prony series creep complance to relaxtion modulus
       using Cholesky decomposition. Using algorithm 4 of "Interconversion of
       linearly viscoelastic material functions expressed as Prony series"

    Parameters
    ----------
    S0 : numpy.array
        Instantaneous creep modulus
    S_mats : numpy.array
```

```
    Creep modulus coefficient
lambdas : numpy.array
    Creep time constants
coeff_size : int
    The size dim of the the matrix i.e. 6 if 6x6

Returns
-------
C0 : numpy.array
    Equilibrium relaxation
C_mats : numpy.array
    Relaxtion modulus coefficient
rhos : numpy.array
    Relaxation time constants
"""

# Number of coefficents
num_coeff = len(S_mats)

# Final amount of coefficents returned, used to size different variables
final_num_coeff = coeff_size * num_coeff

# Following the step by step formula given by the paper
A1 = S0

A2 = np.linalg.cholesky(lambdas[0] * S_mats[0])

for i in range(1, num_coeff):
    temp_mat = lambdas[i] * S_mats[i]
    temp_mat = np.linalg.cholesky(temp_mat)
    A2 = np.concatenate((A2, temp_mat), 1)

A3 = np.identity(coeff_size, dtype=np.float64)
A3 = lambdas[0]*A3

if num_coeff >= 2:
    for i in range(1, num_coeff):
        A3 = linalg.block_diag(A3, (lambdas[i]*np.identity(coeff_size)))

B_idnet = np.identity(final_num_coeff)

L1 = np.linalg.inv(A1)
L2 = A2.T @ L1
L2 = L2.T
L3 = A3 + A2.T @ L1 @ A2

L3_star, PT = np.linalg.svd(L3)[1:]
L3_star = np.diag(L3_star)

L2_star = PT @ L2.T
L2_star = L2_star.T

# Preallocate a 3d numpy array for the coefficent matrices
C_mats = np.empty((final_num_coeff, coeff_size, coeff_size))
```

```
    # Preallocate a numpy array for the time consts
    rhos = np.zeros(final_num_coeff)

    for m in range(0, final_num_coeff):
        for i in range(0, coeff_size):
            for j in range(0, coeff_size):
                C_mats[m, i, j] = (
                    (L2_star[i, m]*L2_star[j, m]) / L3_star[m, m])

        rhos[m] = L3_star[m, m] / B_idnet[m, m]

    C0 = L1

    for i in range(len(C_mats)):
        C0 = C0 - C_mats[i]

    return (C0, C_mats, rhos)

def CtoS(C0, C_mats, rhos, coeff_size):
    """Converts prony series relaxtion modulus to creep complance
       using Cholesky decomposition. Using algorithm 3 of "Interconversion of
       linearly viscoelastic material functions expressed as Prony series"

    Parameters
    ----------
    C0 : numpy.array
        Equilibrium relaxation
    C_mats : numpy.array
        Relaxtion modulus coefficient
    rhos : numpy.array
        Relaxtion time constants
    coeff_size : int
        The size dim of the the matrix i.e. 6 if 6x6

    Returns
    -------
    S0 : numpy.array
        Instantaneous creep modulus
    S_mats : numpy.array
        Creep modulus coefficient
    lambdas : numpy.array
        Creep time constants
    """

    # Number of coefficents
    num_coeff = len(C_mats)

    # Final amount of coefficents returned, used to size different variables
    final_num_coeff = coeff_size * num_coeff

    # Following the step by step formula given by the paper
    L1 = C0
```

```
    for i in range(num_coeff):
        L1 = L1 + C_mats[i]

    L2 = np.linalg.cholesky(rhos[0] * C_mats[0])

    for i in range(1, num_coeff):
        temp_mat = np.linalg.cholesky(rhos[i] * C_mats[i])
        L2 = np.concatenate((L2, temp_mat), 1)

    L3 = np.identity(coeff_size, dtype=np.float64)
    L3 = rhos[0]*L3

    if num_coeff >= 2:
        for i in range(1, num_coeff):
            L3 = linalg.block_diag(L3, (rhos[i]*np.identity(coeff_size)))

    B_idnet = np.identity(final_num_coeff)

    A1 = np.linalg.inv(L1)
    A2 = L2.T @ A1
    A2 = A2.T
    A3 = L3 - L2.T @ A1 @ L2

    A3_star, PT = np.linalg.svd(A3)[1:]
    A3_star = np.diag(A3_star)

    A2_star = PT @ A2.T
    A2_star = A2_star.T

    S0 = A1

    # Preallocate a 3d numpy array for the coefficent matrices
    S_mats = np.empty((final_num_coeff, coeff_size, coeff_size))

    # Preallocate a numpy array for the time consts
    lambdas = np.zeros(final_num_coeff)

    for m in range(0, final_num_coeff):
        for i in range(0, coeff_size):
            for j in range(0, coeff_size):
                S_mats[m, i, j] = (
                    (A2_star[i, m] * A2_star[j, m]) / A3_star[m, m])

        lambdas[m] = A3_star[m, m] / B_idnet[m, m]

    lambdas = np.flip(lambdas)

    return (S0, S_mats, lambdas)

def modulus_at_time(M0, M_mats, time_const, time, property):
    """Gives the matrix of creep or relaxation at a given time

    Parameters
    ----------
```

```
    M0 : numpy.array
        Instantaneous/Equilibrium modulus
    M_mats : numpy.array
        Coefficient moduli
    time_const : numpy.array
        Time constants
    time : float
        Time at which to calculate property
    property : string
        Switch for creep or relaxation calculations (relax or creep)

    Returns
    -------
    mod_time : numpy.array
        Modulus at a given time
    """

    # Number of coefficient matrices
    num_coeff = len(M_mats)

    # Funtion for matrix relaxation modulus at given time
    def relax_time(M_mats, rhos, time): return M_mats * \
        (np.exp(-1 * time * rhos.reshape(num_coeff, 1, 1)))
    # Funtion for matrix creep modulus at given time
    def creep_time(M_mats, lambdas, time): return M_mats * \
        (1 - np.exp(-1 * time * lambdas.reshape(num_coeff, 1, 1)))

    # Sets the proper relax or creep function to the variable time_func
    if property == "relax":
        time_func = relax_time
    elif property == "creep":
        time_func = creep_time
    else:
        raise Exception('Expected "relax" or "creep" proptery')

    # Caluclates the modulus at the given time for each matrix
    mod_time = time_func(M_mats, time_const, time)
    # Sums each matrix together
    mod_time = np.sum(mod_time, axis=0)
    # Adds the Instantaneous/Equilibrium modulus
    mod_time = mod_time + M0

    return mod_time

def nearestPD(A):
    """Find the nearest positive-definite matrix to input

    A Python/Numpy port of John D'Errico's `nearestSPD` MATLAB code [1], whic
h
    credits [2].

    [1] https://www.mathworks.com/matlabcentral/fileexchange/42885-nearestspd

    [2] N.J. Higham, "Computing a nearest symmetric positive semidefinite
```

```
    matrix" (1988): https://doi.org/10.1016/0024-3795(88)90223-6

    Parameters
    ----------
    A : numpy.array
        Matrix

    Returns
    -------
    A3: numpy.array
        Pos def matrix
    """

    B = (A + A.T) / 2
    _, s, V = la.svd(B)

    H = np.dot(V.T, np.dot(np.diag(s), V))

    A2 = (B + H) / 2

    A3 = (A2 + A2.T) / 2

    if isPD(A3):
        return A3

    spacing = np.spacing(la.norm(A))

    Ident = np.eye(A.shape[0], dtype=np.float64)
    k = 1
    while not isPD(A3):
        mineig = np.min(np.real(la.eigvals(A3)))
        A3 += Ident * (-mineig * k**2 + spacing)
        k += 1

    return A3

def isPD(B):
    """Returns true when input is positive-definite, via Cholesky

    Parameters
    ----------
    B : numpy.array
        Matrix

    Returns
    -------
    Bool : Bool
        Returns True or False
    """

    try:
        _ = la.cholesky(B)
        return True
    except la.LinAlgError:
```

```
        return False

def pos_def_update(M0, M_mats):
    """Checks a matrices for positive-definitness. If matrix is not
    positive-definite, finds nearest positive-definite matrix and replaces

    Parameters
    ----------
    M0 : numpy.array
        Instantaneous/Equilibrium modulus
    M_mats : numpy.array
        Coefficient moduli

    Returns
    -------
    M0 : numpy.array
        Positive-definite Instantaneous/Equilibrium modulus
    M_mats : numpy.array
        Positive-definite Coefficient moduli
    """

    def is_pos_def(matrix):
        if isPD(matrix):
            return matrix
        else:
            return nearestPD(matrix)

    M0 = is_pos_def(M0)

    for i in range(len(M_mats)):
        M_mats[i] = is_pos_def(M_mats[i])

    return (M0, M_mats)

def convolution_check_mat(C0, C_mats, rhos, S0, S_mats, lambdas, t):
    """Checks the convolution intergral of the C(t) and S(t) matrices.
       Should be the identiy matrix of t*I
       Also returns the errors from scipy.quad intergration

    Parameters
    ----------
    C0 : numpy.array
        Equilibrium relaxation
    C_mats : numpy.array
        Relaxtion modulus coefficient
    rhos : numpy.array
        Relaxation time constants
    S0 : numpy.array
        Instantaneous creep modulus
    S_mats : numpy.array
        Creep modulus coefficient
    lambdas : numpy.array
        Creep time constants
    t : float
```

```
    time

Returns
-------
convolution : numpy.array
    Numpy array of the convolution matrix for C(t) and S(t)
error : numpy.array
    Numpy array of the errors for each element of the convolution matrix
"""

dim = max(C0.shape)

convolution = np.empty(C0.shape)
error = np.empty(C0.shape)

def f(C0, C_mats, S0, S_mats, lambdas, rhos, t):
    def g(tau):

        C_converted = modulus_at_time(
            C0, C_mats, rhos, t - tau, "relax")

        S_converted = modulus_at_time(
            S0, S_mats, lambdas, tau, "creep")

        return np.dot(C_converted, S_converted)[i, j]
    return g

u = f(C0, C_mats, S0, S_mats, lambdas, rhos, t)

for i in range(dim):
    for j in range(dim):

        convolution[i, j], error[i, j] = integrate.quad(
            u, 0, t, epsabs=1e-12, epsrel=1e-12, limit=1000)

return convolution, error
```

LIST OF REFERENCES

Higham, N. J. (1988). COMPUTING A NEAREST SYMMETRIC POSITIVE SEMIDEFINITE MATRIX. In *Linear Algebra and Its Applications* (Vol. 103, pp. 103–118). https://doi.org/10.1016/0024-3795(88)90223-6

Knauss, W. G., & Zhao, J. (2007). Improved relaxation time coverage in ramp-strain histories. *Mechanics of Time-Dependent Materials, 11*(3), 199–216. https://doi.org/10.1007/s11043-007-9035-4

Lee, S., & Knauss, W. G. (2000). A Note on the Determination of Relaxation and Creep Data from Ramp Tests. *Mechanics of Time-Dependent Materials*, *4*(1), 1–7. https://doi.org/10.1023/A:1009827622426

Li, J., Kwok, K., & Pellegrino, S. (2016). Thermoviscoelastic models for polyethylene thin films. In *Mechanics of Time-Dependent Materials* (Vol. 20, Issue 1, pp. 13–43). https://doi.org/10.1007/s11043-015-9282-8

Luk-Cyr, J., Crochon, T., Li, C., & Levesque, M. (2013). Interconversion of linearly viscoelastic material functions expressed as Prony series: A closure. In *Mechanics of Time-Dependent Materials* (Vol. 17, Issue 1, pp. 53–82). https://doi.org/10.1007/s11043-012-9176-y

Luk-Cyr, Jacques, Crochon, T., Li, C., & Lévesque, M. (2013). Interconversion of linearly viscoelastic material functions expressed as Prony series: A closure. *Mechanics of Time-Dependent Materials*, *17*(1), 53–82. https://doi.org/10.1007/s11043-012-9176-y

Park, S. W., & Schapery, R. A. (1999). Methods of interconversion between linear viscoelastic material functions. Part I—a numerical method based on Prony series. *International Journal of Solids and Structures*, *36*(11), 1653–1675. https://doi.org/10.1016/S0020-7683(98)00055-9

Sorvari, J., & Malinen, M. (2007). Numerical interconversion between linear viscoelastic material functions with regularization. In *International Journal of Solids and Structures* (Vol. 44, Issues 3–4, pp. 1291–1303). https://doi.org/10.1016/j.ijsolstr.2006.06.029