8 Viscoplasticity Models

Chapter Outline

Mechanics of Solid Polymers.
DOI: http://dx.doi.org/10.1016/B978-0-323-31150-2.00008-X

8.1 Introduction

Viscoplasticity models are the ultimate models for polymer modeling, both in terms of accuracy and complexity. Some of the models of this category are truly impressive at predicting the non-linear, time- and temperature-dependent response of various polymers. The downsides of these models are that they typically require more experimental data for proper calibration, they can be numerically expensive, and typically require additional software components (user subroutines, like UMATs and VUMATs, see Chapter 10).

The field of viscoplastic constitutive modeling is evolving, and new models are developed every year by different research groups. This chapter attempts to present some of the most useful models that are available for different polymers, and the trends of where this field is going in the future.

8.2 Bergström-Boyce Model

The Bergström-Boyce (BB) model [1–5] is an advanced constitutive model for predicting the non-linear time-dependent, large-strain behavior of elastomer-like materials. The model has been shown to be accurate for both traditional engineering rubbers, and soft biomaterials [1–5]. The BB-model can also be considered a powerful generalization of linear viscoelasticity, and as will be shown in this section can overcome some of the main issues with linear viscoelasticity related to both large strain deformations and strain amplitude dependence of the dynamic properties (E' and E'').

The motivation for the BB-model can be described using Figure 8.1. This figure shows experimental data for a chloroprene rubber tested in uniaxial compression at two different strain rates (-0.1/s, and -0.03/s), although the qualitative response shown in this figure is true for all elastomers. The red solid curve shown in the figure is the experimentally determined stress-strain response after being exposed to the applied strain history shown in Figure 8.2. The dashed blue line is the experimental data from

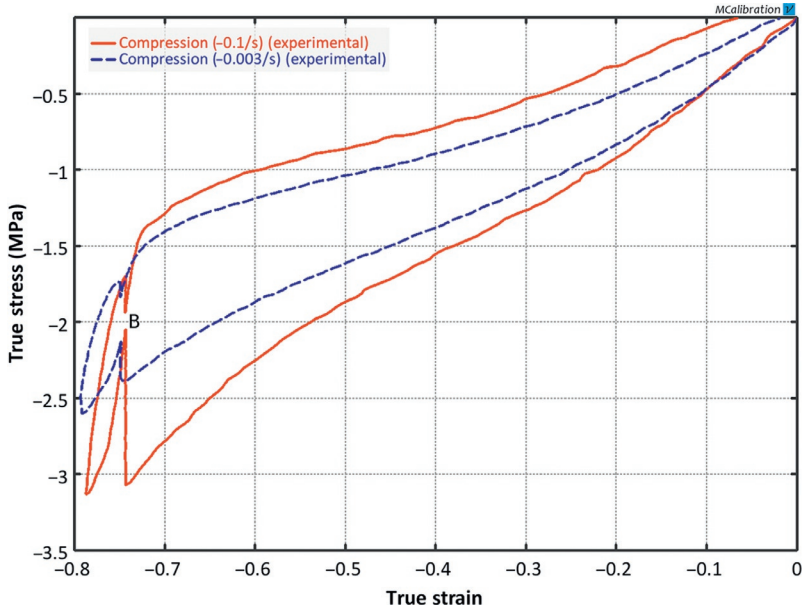


Figure 8.1 Experimental data for a chloroprene rubber tested at two strain rates. The strain was held constant for 10 min at a true strain of −0.75.

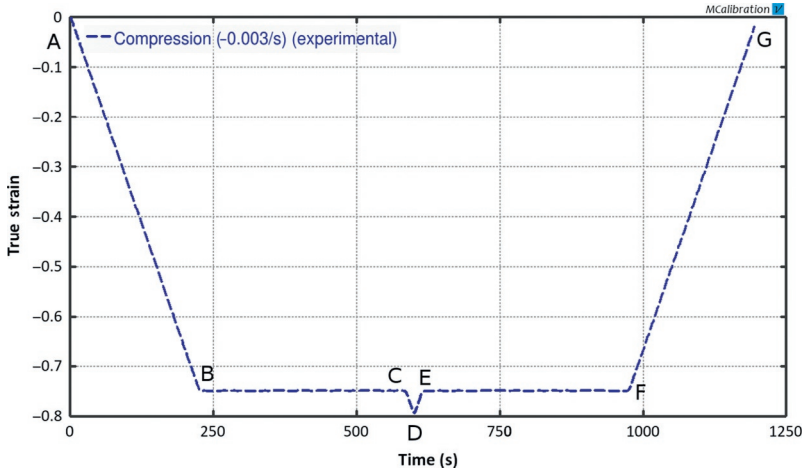


Figure 8.2 Applied strain history used to test the chloroprene rubber. The strain was held constant for 10 min at a true strain of −0.75.

a similar experiment performed at a 50 times slower applied strain-rate. What makes the results in Figure 8.1 so interesting is the stress relaxation that occurs during the hold segments (i.e. B-C and E-F in Figure 8.2). In this case the stress relaxes significantly and the stress relaxation during loading and during unloading appears to asymptotically approach the same stress value. Note that the stress relaxation during unloading is negative. This is, even though the specimen has been compressed and has a negative stress, the stress actually grows larger in magnitude during the stress relaxation segment.

The stress value that is approached at Point B in Figure 8.1 can therefore be considered the *equilibrium stress* at a strain value of about -0.75. By performing a set of experiments of this type, where stress relaxation segments have been inserted, makes it possible to experimentally establish an equilibrium response for the elastomer, see the red curve in Figure 8.3. The equilibrium curve is the hypothetical stress-strain curve that would be

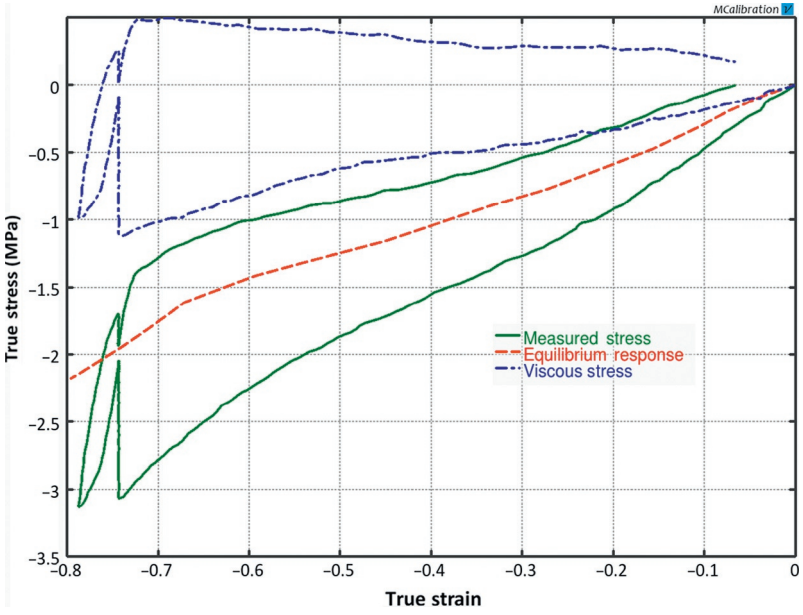


Figure 8.3 The experimentally measured stress (red solid line) can be decomposed into an equilibrium stress (green dashed line) and a viscoelastic stress (blue dash-dot line).

obtained if the experiment was run in infinite time (i.e. if the strain-rate was zero). This observation makes it possible to divide the experimentally determined stress response into an equilibrium response (green dashed line in Figure 8.3), and a time-dependent deviation from the equilibrium response (see the blue dash-dot curve in Figure 8.3).

In other words, the true response of the elastomer can be represented using two parallel networks A and B, see Figure 8.4. Network A is a non-linear hyperelastic network, and network B consists of a non-linear hyperelastic component in series with a non-linear viscoelastic flow element.

Note that the viscoelastic flow response has to be non-linear since otherwise the model framework will be the same as linear viscoelasticity with a one-term Prony series, which was shown in Chapter 6 to be insufficient for capturing the general response of elastomers.

The idea to decompose the total stress into an elastic and a history dependent component was proposed by Green and Tobolsky [6], and the approach to model elastomers as two interacting networks has been used in different variations by Johnson and

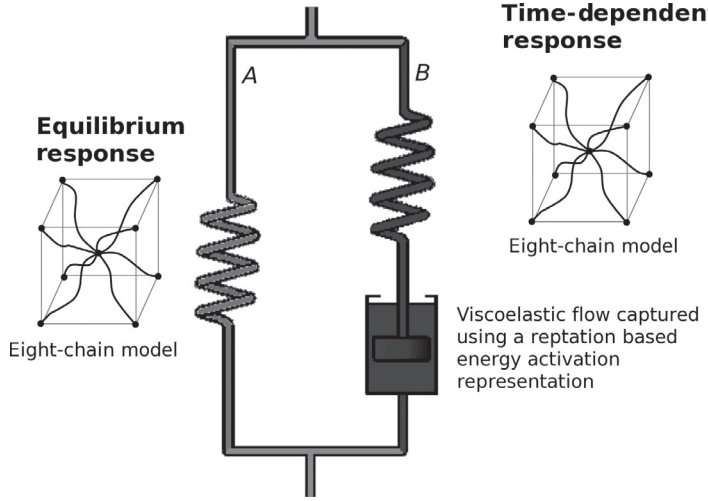


Figure 8.4 Constitutive representation of the elements of the BB model.

Quigley [7], Johnson and Stacer [8], Johnson et al. [9], Roland [10], and Roland and Warzel [11].

In the BB model the applied deformation gradient is acting on two parallel macromolecular networks: $\mathbf{F} = \mathbf{F}_A = \mathbf{F}_B$. The deformation gradient acting on the time-dependent network B is further decomposed into elastic and viscoelastic components:

$$\mathbf{F}_B = \mathbf{F}_B^e \mathbf{F}_B^v. \tag{8.1}$$

The decomposition of the deformation gradient into elastic and viscoelastic components is shown in the deformation map in Figure 8.5.

The response of network A is given by the eight-chain model (see Section 5.3.10):

$$\boldsymbol{\sigma}_A = \frac{\mu}{J\overline{\lambda^*}} \frac{\mathcal{L}^{-1}\left(\overline{\lambda^*}/\lambda^{\text{lock}}\right)}{\mathcal{L}^{-1}\left(1/\lambda^{\text{lock}}\right)} \text{dev}[\mathbf{b}^*] + \kappa(J-1)\mathbf{I}, \tag{8.2}$$

where $[\mu, \lambda^{\text{lock}}, \kappa]$ are material parameters, $\mathcal{L}^{-1}(\cdot)$ the inverse Langevin function, $J = \det[\mathbf{F}]$, and $\mathbf{b}^*$ the distortional part of the left Cauchy-Green tensor. The stress on network B is also given by the eight-chain model, but with a different effective shear modulus:

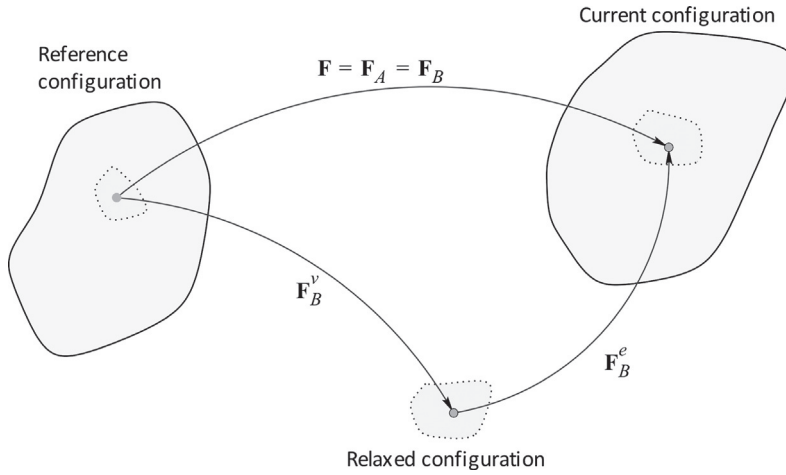


Figure 8.5 Multiplicative decomposition of the deformation.

$$\boldsymbol{\sigma}_B = \frac{s\,\mu}{J_B \overline{\lambda_B^{e*}}} \frac{\mathcal{L}^{-1}\left(\overline{\lambda_B^{e*}}/\lambda^{\text{lock}}\right)}{\mathcal{L}^{-1}\left(1/\lambda^{\text{lock}}\right)} \operatorname{dev}[\mathbf{b}_B^{e*}] + \kappa(J_B^e - 1)\mathbf{I}, \tag{8.3}$$

where s is a dimensionless material parameter specifying how much stiffer the shear modulus of network B is relative to network A, and $\overline{\lambda_B^{e*}}$ is the chain stretch in the elastic part of Network B. Using this representation the total Cauchy stress is given by

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_A + \boldsymbol{\sigma}_B. \tag{8.4}$$

Note that the presentation here is based on the original BB-model in which the hyperelastic response is given by the Arruda-Boyce (AB) eight-chain model. This approach is also used in the ANSYS [12], MSC.Marc [13], and LS-DYNA [14] native implementations of the BB-model. The Abaqus [15] implementation (called `*HYSTERESIS`), is more general and allows for any of the built-in hyperelastic models to be used.

Equations (8.1)–(8.3) completely specify how the stress can be calculated in any given deformation state (with known internal state $\mathbf{F}_B^v$). To complete the model it is also necessary to specify the rate of change in the internal state of the material through viscoelastic flow, that is, the rate kinematics of the model. The velocity gradient on network B, $\mathbf{L}_B = \dot{\mathbf{F}}_B \mathbf{F}_B^{-1}$, can be decomposed into elastic and viscous components as follows:

$$\begin{aligned}
\mathbf{L}_B &= \left[\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{F}_B^e \mathbf{F}_B^v\right)\right]\left(\mathbf{F}_B^e \mathbf{F}_B^v\right)^{-1} \\
&= \left[\dot{\mathbf{F}}_B^e \mathbf{F}_B^v + \mathbf{F}_B^e \dot{\mathbf{F}}_B^v\right]\left(\mathbf{F}_B^v\right)^{-1}\left(\mathbf{F}_B^e\right)^{-1} \\
&= \dot{\mathbf{F}}_B^e \left(\mathbf{F}_B^e\right)^{-1} + \mathbf{F}_B^e \dot{\mathbf{F}}_B^v \left(\mathbf{F}_B^v\right)^{-1}\left(\mathbf{F}_B^e\right)^{-1} \\
&= \mathbf{L}_B^e + \mathbf{F}_B^e \mathbf{L}_B^v \mathbf{F}_B^e \\
&= \mathbf{L}_B^e + \tilde{\mathbf{L}}_B^v,
\end{aligned} \tag{8.5}$$

where

$$\mathbf{L}_B^v = \dot{\mathbf{F}}_B^v \left(\mathbf{F}_B^v\right)^{-1} = \mathbf{D}_B^v + \mathbf{W}_B^v, \tag{8.6}$$

$$\tilde{\mathbf{L}}_B^v = \tilde{\mathbf{D}}_B^v + \tilde{\mathbf{W}}_B^v. \tag{8.7}$$

To make the unloading unique [16], prescribe $\tilde{\mathbf{W}}_B^v \equiv 0$. The rate of viscous deformation of network B is constitutively prescribed by:

$$\tilde{\mathbf{D}}_B^v = \dot{\gamma}_B(\boldsymbol{\sigma}_B, \mathbf{b}_B^{e*})\, \mathbf{N}_B^v, \tag{8.8}$$

where

$$\mathbf{N}_B^v = \frac{\text{dev}[\boldsymbol{\sigma}_B]}{\tau} = \frac{\text{dev}[\boldsymbol{\sigma}_B]}{||\,\text{dev}[\boldsymbol{\sigma}_B]||_F} \tag{8.9}$$

provides the direction of the viscoelastic flow, and τ is the effective (scalar) stress driving the viscous flow.

The time derivative of $\mathbf{F}_B^v$ can be derived as follows:

$$\tilde{\mathbf{L}}_B^v = \dot{\gamma}_B^v \mathbf{N}_B^v, \tag{8.10}$$

$$\Rightarrow \quad \mathbf{F}_B^e \dot{\mathbf{F}}_B^v \left(\mathbf{F}_B^v\right)^{-1} \left(\mathbf{F}_B^e\right)^{-1} = \dot{\gamma}_B^v \mathbf{N}_B^v,$$

$$\Rightarrow \quad \dot{\mathbf{F}}_B^v = \dot{\gamma}_B^v \left(\mathbf{F}_B^e\right)^{-1} \frac{\text{dev}[\boldsymbol{\sigma}_B]}{||\,\text{dev}[\boldsymbol{\sigma}_B]||_F} \mathbf{F}_B^e \mathbf{F}_B^v. \tag{8.11}$$

These equations for the time-derivative of the deformation gradient are based on pure kinematics, and only embody the physics of a specific material through a scalar equation for the effective flow rate $\dot{\gamma}_B$. The effective flow rate $\dot{\gamma}_B$ of Network B must be constitutively prescribed. Here, a micromechanism-inspired model is proposed on the assumption that the mechanism responsible for the time-dependent behavior is the reptation of macromolecules that are "elastically inactive" (i.e. molecules that carry less load and have the capability to significantly change conformation during creep loading.)

To illustrate this view consider first an over-simplified model with one free chain located in a network of chains as shown in Figure 8.6.

If the network is deformed at a high enough rate then the free chain will also deform more or less affinely with the network. Hence the entropy of the free chain is decreased and the free chain contributes additional deformation resistance. If the applied strain is then held constant in the deformed state the free chain will slowly, by Brownian motion, return to a more relaxed

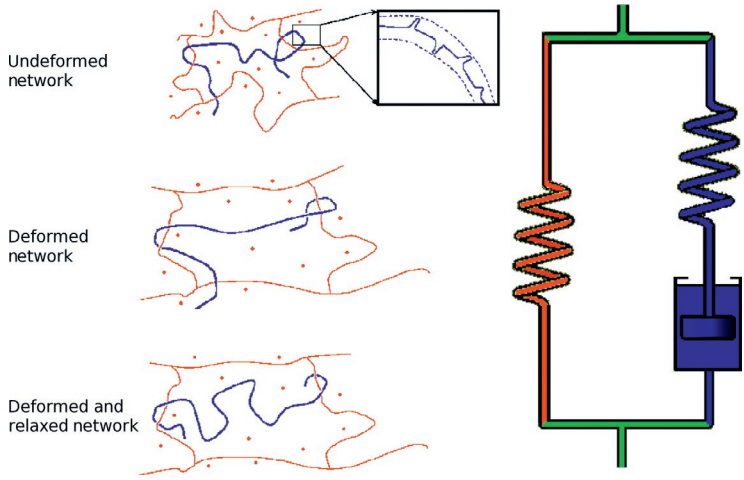


Figure 8.6 Schematic representation of the molecular deformation that occurs during deformation and stress relaxation of crosslinked polymer.

configuration. The rate of return toward a relaxed configuration is given by the governing reptation motion of the free chain [17].

Let us now turn our attention to a more realistic situation in which free chains do not exist. In a real polymer network, however, there are almost always free chain-ends which behave as the free chain described above. A reasonable extension to the free chain and free chain-end models is to consider also inactive chain segments such as $A - B - C$ illustrated in Figure 8.7.

Qualitatively, the same behavior is exhibited by the inactive chain segment in Figure 8.7 and the free chain in Figure 8.6. The loop $A - B - C$ in Figure 8.7 undergoes Brownian motion and has an equilibrium position at a finite distance from the constraining chain DD'. Hence, DD' behaves as an obstacle which imposes an energy barrier to the relaxation process. The relaxation toward equilibrium can therefore be considered to be energy activated.

To develop the constitutive equation for the time-dependent element consider a free chain-end of the type illustrated in Figure 8.7, but bear in mind that the presented arguments also hold for the more general situation shown in Figure 8.6.

The chain segment at B is constrained to travel back and forth along the constraining tube by Brownian motion in a combination

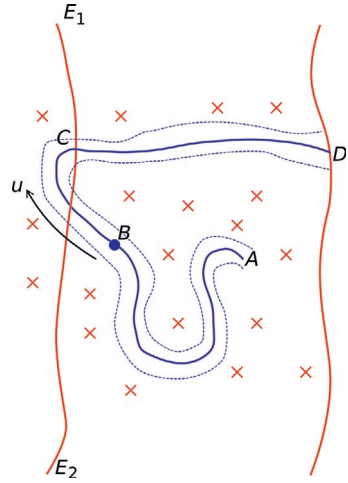


Figure 8.7 The stress relaxation is driven by molecular reptation.

of reptation motion and contour length fluctuations. Call the displacement of B along the tube u. The average displacement of B can be shown by the theory of reptational motion of chain molecules [18] to scale as $\langle u \rangle = C_3 \langle u^2 \rangle^{1/2} \equiv C_3 \sqrt{\phi(t)}$. The effective length of the chain in a creep experiment can consequently be written

$$l(t) = l_0 + C_3 \sqrt{\phi(t)}, \tag{8.12}$$

where the mean square displacement of the chain segment $\phi(t)$ has been derived [18] by reptation dynamics considerations to scale as

$$\phi(t) \propto \begin{cases} t^{1/2}, & t \leq \tau_e, \\ t^{1/4}, & \tau_e \leq t \leq \tau_R, \\ t^{1/2}, & \tau_R \leq t \leq \tau_d, \\ t, & \tau_d \leq t, \end{cases} \tag{8.13}$$

where τ_e is the time at which the tube constraint is first felt, τ_R is the Rouse relaxation time, and τ_d is the tube disengagement time. Equations (8.12) and (8.13) can be simplified to

$$\lambda^{Bp}_{\text{chain}}(t) = \frac{l(t)}{l_0} = 1 + C_4 t^{C_5}, \tag{8.14}$$

where $C_4 > 0$ and $C_5 \in [0.5, 1.0]$. Taking the time derivative of (8.14) gives

$$\dot{\lambda}^{Bp}_{\text{chain}} = C_4 C_5 t^{C_5 - 1}. \tag{8.15}$$

It is now possible to eliminate t between (8.14) and (8.15) giving

$$\dot{\lambda}^{Bp}_{\text{chain}} = C_6 \left[\lambda^{Bp}_{\text{chain}} - 1 \right]^{C_7}, \tag{8.16}$$

where $C_6 > 0$ and C_7 is about -1. This equation shows how the *effective* creep rate depends on the chain stretch, where the chain stretch is correlated to the principal macroscopic stretch state by the 8-chain assumption [19], i.e.

$$\lambda^{Bp}_{\text{chain}} = \sqrt{\frac{I_1^{Bp}}{3}} = \sqrt{\frac{\left(\lambda_1^{Bp}\right)^2 + \left(\lambda_2^{Bp}\right)^2 + \left(\lambda_3^{Bp}\right)^2}{3}}. \tag{8.17}$$

Equation (8.16) gives the creep rate at a constant stress level, but the creep rate also depends on the level of the applied stress. And as discussed above, this stress dependence is assumed to be energy activated. The microstructural connection to the activation parameters is complicated; it is possible, however, to use a generic expression of the form

$$\dot{\gamma}_B = C_1 \left[\lambda^{Bp}_{\text{chain}} - 1 + \xi \right]^{C_2} \left(\frac{\tau_B}{\hat{\tau}_B} \right)^m, \tag{8.18}$$

where τ_B is the effective stress measure introduced in Equation (8.18), $\hat{\tau}_B$ is a material constant, and ξ is small positive constant that is introduced to eliminate a removable singularity in the flow rate in the undeformed state [4].

Note that in the proposed model the constants $\hat{C}_1 \equiv C_1 / \hat{\tau}_B^m$ and m are positive; and C_2 is a constant that is restricted by reptational dynamics to be negative.

In summary, the rate-equation for viscous flow is given by [1]:

$$\dot{\gamma}^v_B = \dot{\gamma}_0 \left(\overline{\lambda^v_B} - 1 + \xi \right)^C \left[R \left(\frac{\tau}{f_v \tau_{\text{base}}} - \hat{\tau}_{\text{cut}} \right) \right]^m, \tag{8.19}$$

where:

- $[\xi, C, \tau_{\text{base}}, \hat{\tau}_{\text{cut}}, m]$ are material parameters.
- $\dot{\gamma}_0 \equiv 1/\text{s}$ is a constant introduced to ensure dimensional consistency.
- $R(x) = (x + |x|)/2$ is the ramp function.
- $f_v = 1 + \alpha \boldsymbol{\varepsilon} : \boldsymbol{\varepsilon}_e$ is a network interaction factor. This is a modification of the original BB flow model [3] that makes the flow resistance stress dependent on the network state through a factor f_v that depends on the two strains $\boldsymbol{\varepsilon} = \ln[\mathbf{v}]$, and $\boldsymbol{\varepsilon}_e = \ln[\mathbf{v}_e]$. The reason for the modification is that most elastomer-like materials are experimentally shown to have less strain-rate dependence during unloading than during loading. This flow model introduces the ability to capture this response.
- $\hat{\tau}_{\text{cut}}$ is a cut-off stress below which no flow will occur. The ramp function is introduced in order to increase the numerical efficiency of the material model for cases when regions of the FE mesh is not undergoing significant deformations, and to allow for predictions of true plastic deformation.
- $\overline{\lambda_B^v} = \sqrt{\frac{\text{tr}[\mathbf{b}_B^v]}{3}}$ is the viscoelastic chain stretch.
- The effective stress driving the viscous flow is:

$$\tau = ||\,\text{dev}[\boldsymbol{\sigma}_B]||_F = \sqrt{\text{tr}\left[\boldsymbol{\sigma}'_B \boldsymbol{\sigma}'_B\right]}, \tag{8.20}$$

where $||\cdot||_F$ is the Frobenius norm.

8.2.1 Matlab Implementation of the BB-Model

As was discussed in Section 8.2, the BB material model is formulated as a set of differential equations that need to be solved for each time increment. One way to solve this set of equations is to use the following algorithm:

1. Known values at time t_i:
   - Deformation gradient: $\mathbf{F}$
   - State variables: $\mathbf{F}_B^v$

2. Known values at time t_{i+1}:
    - Deformation gradient: $\mathbf{F}$
3. Calculate $\mathbf{F}_B^v$ at t_{i+1} using an ODE solver and Equation (8.11) for $\dot{\mathbf{F}}_B^v(t, \mathbf{F}_B^v)$.
4. Calculate $\mathbf{F}^e$ using Equation (8.1), and σ_B using Equation (8.3) at t_{i+1}.

For the case of incompressible, uniaxial loading this algorithm can be implemented in Matlab as shown in Figure 8.8.

Matlab File Name: mat_BB.m

```
function [stress] = mat_BB(time, strain, params)
%mat_BB Bergstrom-Boyce model
%Incompressible uniaxial loading
%This function is using true stress and strain
muA     = params(1);
lambdaL = params(2);
sB      = params(3);
xi      = params(4);
C       = params(5);
tauHat  = params(6);
m       = params(7);
stress = 0 * strain;
lambdaP = 1;
for i = 2 : length(strain)
    stressA = mat_EC(time(i), strain(i), [muA lambdaL]);
    [t,lambdaP] = ode45(@(t,y) lambdaPDot(t, y, time, strain, sB*muA, lambdaL, ...
        xi, C, tauHat, m), time(i-1:i), lambdaP); % find lambdaP at t+dt
    lambdaP = lambdaP(end);
    strainP = log(lambdaP);
    strainE = strain(i) - strainP;
    stressB = mat_EC(time(i), strainE, [muA*sB lambdaL]);
    stress(i) = stressA + stressB;
end
end

function lambdaPDot = lambdaPDot(time, lambdaP, timeVec, strainVec, muB, lambdaL, ...
    xi, C, tauHat, m)
lambda = exp( interp1(timeVec, strainVec, time) ); % stretch at the specified time
lambdaE = lambda / lambdaP;
lambdaPChain = sqrt((lambdaP^2 + 2 / lambdaP) / 3);
stressB = mat_EC(time, log(lambdaE), [muB lambdaL]);
gamDot = (lambdaPChain - 1 + xi)^C * (abs(stressB) / tauHat)^m;
lambdaPDot = gamDot * lambdaE * sign(stressB) * lambda;
end
```

Matlab File Name: test_mat_BB.m

```
time = linspace(0,2)';
strain = [linspace(0, 0.5,50)'; linspace(0.5, 0, 50)'];
stress = mat_BB(time, strain, [1 4 3 0.05 -0.5 1.0 4.0]);
plot(strain,stress)
```

Figure 8.8 (a) Matlab implementation of the Bergström-Boyce model. (b) Exemplar Matlab code that calls the Matlab implementation of the BB model.

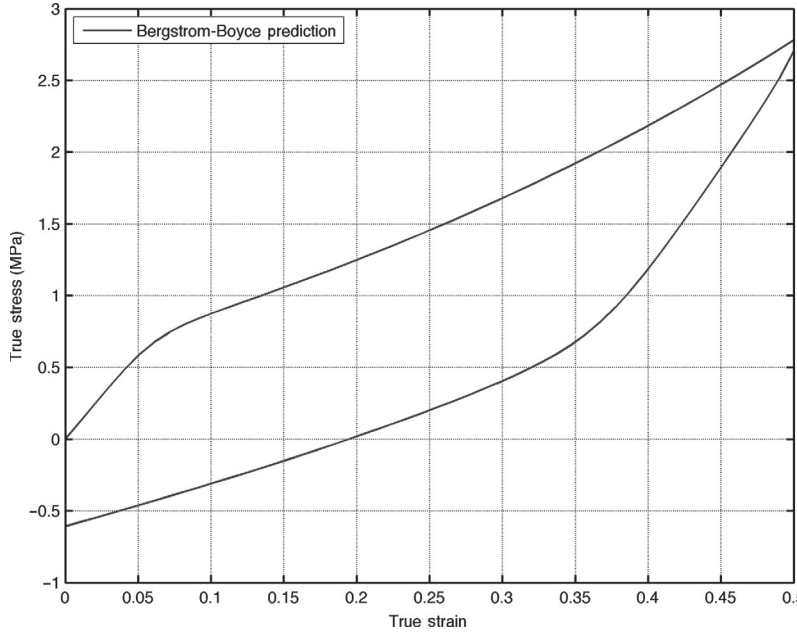


Figure 8.9 Predictions from the Matlab file `test_mat_BB.m`.

The results from running the test file `test_mat_BB` are shown in Figure 8.9.

8.2.2 Python Implementation of the BB-Model

For incompressible uniaxial loading the BB model can be implemented into Python code as shown below. The function `uniaxial_stress_visco()` is a generic function for calculating the stress for any viscoplastic material model specified by the function `model()`. This function searches for the transverse strain, at each time increment, that gives the correct stress boundary condition ($\sigma_{22} = \sigma_{33} = 0$).

The implementation of the BB-model is provided in two functions `BB_timeDer_3D()` and `BB_3D()`. The time-derivative function is used by the Python function `scipy.integrate.odeint()`.

Code to "Polymer_Mechanics_Chap09.py":

```
from Polymer_Mechanics_Chap05 import *
import scipy.integrate

def ramp(x):
    return (x + abs(x)) / 2.0

def toVec(A):
    """Convert a 3x3 matrix to vector"""
    return array([A[0][0], A[1][1], A[2][2]])

def uniaxial_stress_visco(model, timeVec, trueStrainVec, params):
    """Compressible uniaxial loading. Returns true stress."""
    stress = zeros(len(trueStrainVec))
    lam2_1 = 1.0
    FBv1 = array([1.0, 1.0, 1.0])
    for i in range(1, len(trueStrainVec)):
        print 'uniaxial_stress: i=', i, ' of ', len(trueStrainVec)
        time0 = timeVec[i-1]
        time1 = timeVec[i]
        lam1_0 = exp(trueStrainVec[i-1])
        lam1_1 = exp(trueStrainVec[i])
        lam2_0 = lam2_1
        F0 = array([lam1_0, lam2_0, lam2_0])
        F1 = array([lam1_1, lam2_1, lam2_1])
        FBv0 = FBv1.copy()
        calcS22Abs = lambda x: abs(model(F0, array([lam1_1,x,x]), \
           FBv0, time0, time1, params)[0][1])
        # search for transverse stretch that gives S22=0
        lam2_1 = scipy.optimize.fmin(calcS22Abs, x0=lam2_0,
                                     xtol=1e-9, ftol=1e-9, disp=False)
        res = model(F0, array([lam1_1, lam2_1, lam2_1]), FBv0, \
           time0, time1, params)
        stress[i] = res[0][0]
        FBv1 = res[1]
    return stress
```

Additional Code to "Polymer_Mechanics_Chap09.py":

```
def BB_timeDer_3D(Fv, t, params, time0, time1, F0, F1):
    """Returns FvDot"""
    mu, lamL, kappa, s, xi, C, tauBase, m, tauCut = params[:9]
    F = F0 + (t-time0) / (time1-time0) * (F1-F0)
    Fe = F / Fv
    Stress = toVec(EC_3D(Fe, [s*mu, lamL, kappa]))
    devStress = Stress - sum(Stress)/3
    tau = norm(devStress)
    lamCh = sqrt(sum(Fv*Fv)/3.0)
    lamFac = lamCh - 1.0 + xi
    gamDot = lamFac**C * (ramp(tau/tauBase-tauCut)**m)
    prefac = 0.0
    if tau > 0: prefac = gamDot / tau
    FeInv = array([1.0, 1.0, 1.0]) / Fe
    FvDot = prefac * (FeInv * devStress * F)
    return FvDot

def BB_3D(F0, F1, FBv0, time0, time1, params):
    """BB-model. 3D loading specified by principal stretches.
       params: [muA, lamL, kappa, s, xi, C, tauHat, m, tauCut].
       Returns: (true stress, FBv1)"""
    muA, lamL, kappa, s = params[:4]
    StressA = toVec(EC_3D(F1, [muA, lamL, kappa]))
    FBv1 = scipy.integrate.odeint(BB_timeDer_3D, FBv0, \
            [time0, time1], args=(params, time0, time1, F0, F1))[1]
    FBe1 = F1 / FBv1
    StressB = toVec(EC_3D(FBe1, [s*muA, lamL, kappa]))
    Stress = StressA + StressB
    return (Stress, FBv1)
```

The actual commands that sets everything up and calls the main function are listed in the file `BB_Compressible_Uniaxial.py`.

Code "BB_Compressible_Uniaxial.py":

```
from Polymer_Mechanics_Chap05 import *
from Polymer_Mechanics_Chap09 import *

N = 100
timeVec = linspace(0, 10.0, N)
trueStrain = linspace(0, 0.2, N)
params = [2.0, 3.5, 500.0, 3.0, 0.05, -0.5, 0.5, 8.0, 0.01]
trueStress = uniaxial_stress_visco(BB_3D, timeVec, trueStrain, \
   params)

plot(trueStrain, trueStress, 'r-', label='Python Calculation')
xlabel('True Strain')
ylabel('True Stress (MPa)')
grid('on')
show()
```

The results from running this code are shown in Figure 8.10.

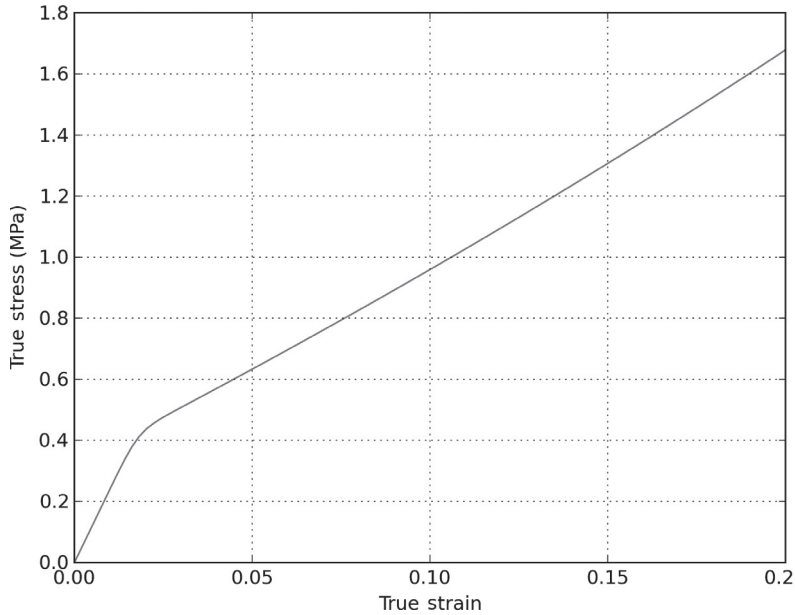


Figure 8.10 Predictions from the Python file `BB_Compressible_Uniaxial.py`.

8.2.3 Generic Numerical Implementation

The theory for the BB model, presented above, used the viscoelastic deformation gradient ($\mathbf{F}_B^v$) as a state variable for amount of

viscoelastic flow. This is not the only possible state variable. In fact, you can write the evolution equation for the viscoplastic deformation in many different and alternative ways:

$$\dot{\mathbf{F}}_v = \frac{\dot{\gamma}}{||\,\mathrm{dev}[\mathbf{b}_e^*]||}\mathbf{F}_e^{-1}\,\mathrm{dev}[\mathbf{b}_e^*]\mathbf{F}, \tag{8.21}$$

$$\dot{\mathbf{C}}_v^* = \frac{2\dot{\gamma}}{||\,\mathrm{dev}[\mathbf{b}_e^*]||}\left[\mathbf{C}^* - \frac{1}{3}\,\mathrm{tr}[\mathbf{b}_e^*]\mathbf{C}_v^*\right], \tag{8.22}$$

$$\dot{\mathbf{E}}_v = \frac{\dot{\gamma}}{||\,\mathrm{dev}[\mathbf{b}_e^*]||}\left[\mathbf{C} - \frac{1}{3}\,\mathrm{tr}[\mathbf{b}_e^*]\mathbf{C}_v\right], \tag{8.23}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\mathbf{F}_v^{-1}\right] = -\dot{\gamma}\mathbf{F}_v^{-1}\frac{\mathrm{dev}[\mathbf{F}_v^{-\top}\mathbf{U}^{*2}\mathbf{F}_v^{-1}]}{||\,\mathrm{dev}[\mathbf{b}_e^*]||}, \tag{8.24}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\mathbf{c}_v^{-1}\right] = \frac{-2\dot{\gamma}}{||\,\mathrm{dev}[\mathbf{b}_e^*]||}\left(\mathbf{c}_v^{-1}\mathbf{U}^{*2} - \overline{\lambda_e^*}^2\mathbf{I}\right)\mathbf{c}_v^{-1}. \tag{8.25}$$

In these equations $\mathbf{C}_v^*$ is the viscoelastic right Cauchy-Green tensor, $\mathbf{E}_v$ is the Green strain, $\mathbf{F}_v^{-1}$ is the inverse of the viscoelastic deformation gradient, and $\mathbf{C}_v^{-1}$ is called the Finger tensor. All of these equation are equally valid, and the most appropriate one can be selected based on personal preference and the desired reference frame to use for the calculations.

8.2.4 Dynamic Loading Predictions

The dynamic response of elastomers and other soft materials is an important characteristic in many industrial applications. One common way to experimentally measure the dynamic response is to apply a sinusoidal strain with a constant strain amplitude (ε_a) and a constant mean strain (ε_m):

$$\varepsilon(t) = \varepsilon_m + \varepsilon_a \sin(\omega t). \tag{8.26}$$

If the strains are sufficiently small then the measured stress response will also be sinusoidal, but it will be out-of-phase with the applied strain:

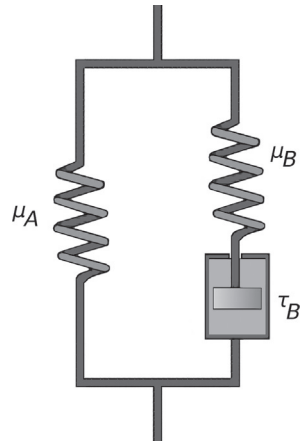
$$\sigma(t) = \sigma_m + \sigma_a \sin(\omega t + \delta), \tag{8.27}$$

$$= \sigma_m + \sigma_a E' \sin(\omega t) + \sigma_a E'' \cos(\omega t). \tag{8.28}$$

From this the storage modulus (E') and the loss modulus (E'') can be defined from:

$$\sigma(t) = \sigma_m + \varepsilon_a E' \sin(\omega t) + \varepsilon_a E'' \cos(\omega t). \tag{8.29}$$

Experimentally, the storage and loss moduli of most elastomers depend on both the applied strain amplitude and the applied frequency. It is therefore often desirable to have a material model that agrees with this experimental fact. If we define a *Linear Flow Model* as a two-network model where Network *A* consists of a Neo-Hookean spring, and Network *B* consists of a Neo-Hookean spring in series with a linear dashpot (see the following figure).



The flow of the linear dashpot is given by

$$\dot{\gamma}_B = \dot{\gamma}_0 \left[\frac{\tau}{\tau_B} \right], \tag{8.30}$$

where $\dot{\gamma}_0 \equiv 0$. For this model the characteristic relaxation time is given by: $\tau_B/(\dot{\gamma}_0 \mu_B)$, and the characteristic frequency is given by: $\dot{\gamma}_0 \mu_B / \tau_B$.

The predicted response for a strain amplitude sweep for this model is shown in Figure 8.11. As expected, the predicted storage

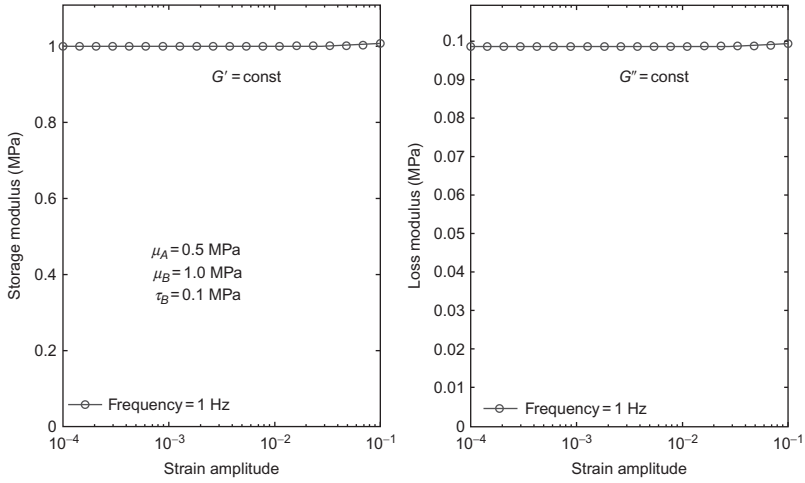


Figure 8.11 Predictions storage and loss moduli for a linear flow model when exposed to a strain amplitude sweep.

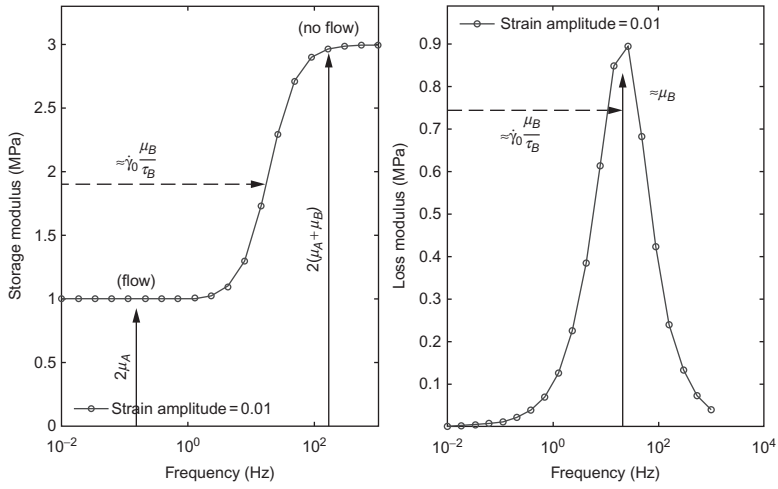


Figure 8.12 Predictions storage and loss moduli for a linear flow model when exposed to a frequency sweep.

and loss moduli are both finite, but independent of the applied strain amplitude.

The predicted response for a frequency sweep for the linear flow model is shown in Figure 8.12. Similar to what was shown for linear viscoelasticity, the storage modulus increases with

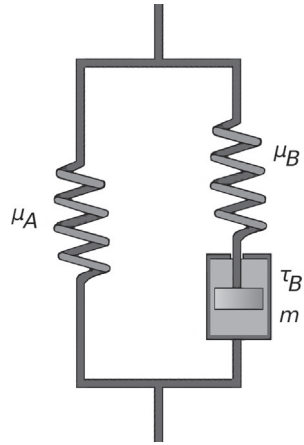
frequency, and the loss modulus has a peak at a characteristic frequency.

As is shown in Figures 8.11 and 8.12, the linear flow model is not sufficiently advanced to capture the true viscoelastic response of many polymers that also depend on the strain amplitude.

An extension of this model, called the *Power Flow Model*, is exactly as the linear flow model except that the viscoelastic flow rate is given by:

$$\dot{\gamma}_B = \dot{\gamma}_0 \left[\frac{\tau}{\tau_B} \right]^m . \tag{8.31}$$

This equation can be represented in the following rheological representation.



This model is a slightly simplified version of the BB model.

The predicted response for a strain amplitude sweep for this model is shown in Figure 8.13. The predicted storage modulus decreases with increasing strain amplitude, this is what is called the *Payne effect*. Similarly, the loss modulus goes through a peak for a given strain amplitude. Both of these behaviors are consistent with what is experienced by typical materials.

The predicted response for a frequency sweep for the power flow model is shown in Figure 8.14. Similar to what was shown for the linear flow model, the storage modulus increases with

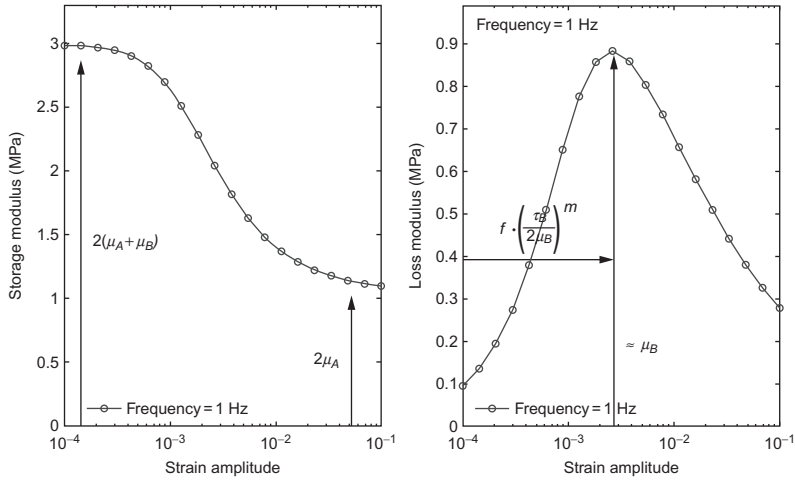


Figure 8.13 Predicted storage and loss moduli for a power flow model when exposed to a strain amplitude sweep.

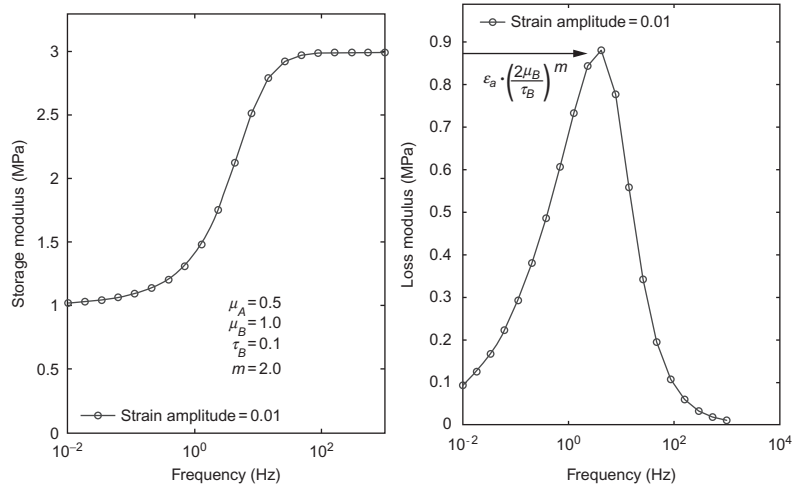


Figure 8.14 Predicted storage and loss moduli for a power flow model when exposed to a frequency sweep.

frequency, and the loss modulus has a peak at a characteristic frequency.

As is shown in Figures 8.13 and 8.14, the power flow model is significantly more accurate compared to the linear flow model when it comes to predicting both the strain amplitude, and fre-

quency dependence. The power flow model (and therefore also the BB model) qualitatively captures all essential dynamic behaviors of elastomers. To capture the response also quantitatively can require multiple parallel networks of the type used in the Power Flow Model. Each network contributing a portion of the strain amplitude and frequency spectrum.

8.2.5 Use of the BB-Model for Polymer Modeling

The BB model is very good at predicting the large-strain time-dependent behavior of isotropic elastomer-like materials. The model is particularly suitable for predicting large-strain, cyclic, stress relaxation, and creep behaviors.

Figure 8.15 shows experimental data for a chloroprene rubber with 25 vol% carbon black. The experimental data consists of loading-unloading cycles at three different strain rates.

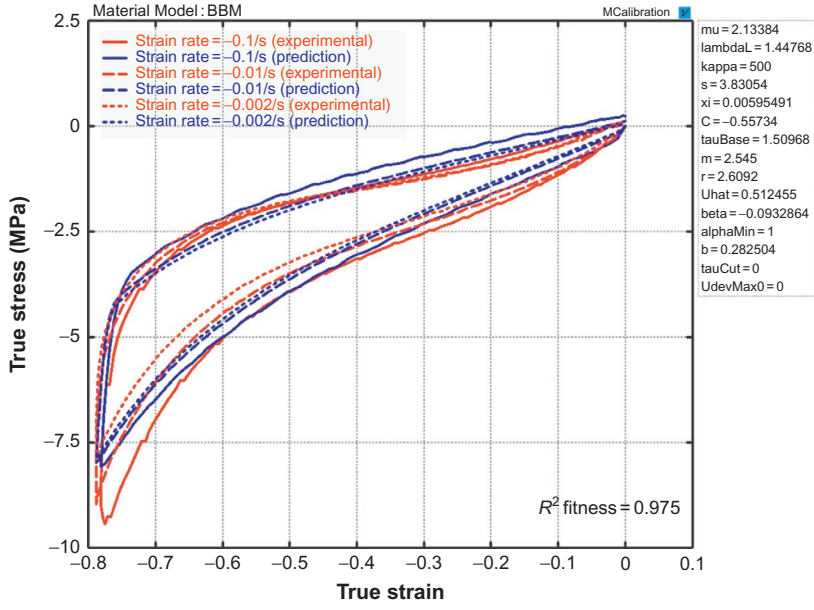


Figure 8.15 Comparison between experimental data for a chloroprene rubber with 25 vol% carbon black, and predictions from the BB-model with Mullins softening.

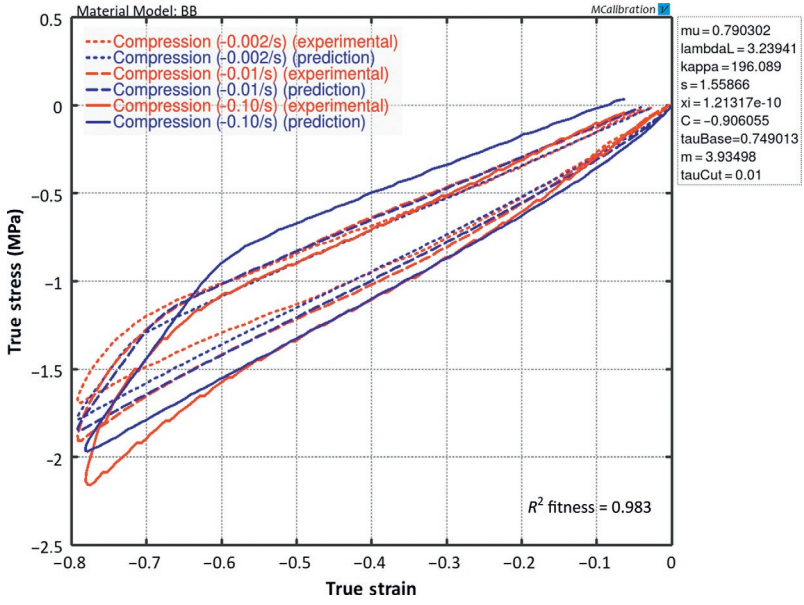


Figure 8.16 Comparison between experimental data for a chloroprene rubber with 7 vol% carbon black, and predictions from the BB-model.

The experiments were performed at room temperature. In the figure is also shown the best fit of the BB-model. It is shown that the model accurately captures the loading-unloading behavior at three strain rates.

Another example, this time for a chloroprene rubber with 7% volume is shown in Figure 8.16.

Also in this case the material was tested in uniaxial compression at three different strain rates. The BB model was calibrated to the experimental data and the model predictions are indicated in the figure. The figure shows that the BB model captures the experimentally observed behavior.

8.3 Arruda-Boyce Model

One of the first advanced models for predicting the response of glassy polymers is the AB model [16, 20, 21]. This model is not the same as the AB eight-chain model which is a hyperelastic model discussed in Section 5.3.10 of Chapter 5. This AB viscoplasticity model is interesting due to its simplicity

and structure, that is also used in more advanced models for thermoplastics.[1]

The AB model was developed [16, 20, 21] for predicting the large strain, time- and temperature-dependent response of glassy polymers. The behavior of this class of materials when subjected to gradually increasing loads is typically characterized by an initial linear elastic response followed by yielding and then strain hardening at large deformations.[2] This evolution in material response with applied loads is directly incorporated into the AB model.

In the AB framework, the total deformation gradient is decomposed into elastic and plastic components, $\mathbf{F} = \mathbf{F}^e\mathbf{F}^p$. As is shown in the one-dimensional rheological representation in Figure 8.17, this decomposition can be interpreted as two networks acting in series: one elastic network (e) and one plastic network (p).

Using this decomposition of the deformation gradient the Cauchy stress can be calculated from the linear elastic relationship:

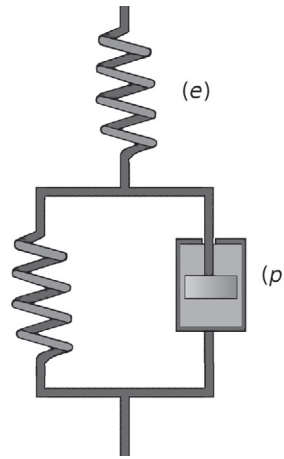


Figure 8.17 Rheological representation of the Arruda-Boyce model.

[1] *The AB model is available in the PolyUMod library [22].*

[2] *Some thermoplastics, for example LDPE, does not have a linear elastic response even at very small strains. The material behaves in a non-linear viscoelastic manner even at small strain.*

$$\mathbf{T} = \frac{1}{J^e}\left(2\mu^e \mathbf{E}^e + \lambda \,\mathrm{tr}[\mathbf{E}^e]\mathbf{I}\right), \tag{8.32}$$

where $\mathbf{E}^e = \ln[\mathbf{V}^e]$ is the logarithmic true strain, $J^e = \det[\mathbf{F}^e]$, and μ^e, λ^e are Lamé's constants. The stress driving the plastic flow is given by the tensorial difference between the total stress and the convected back stress

$$\mathbf{T}^* = \mathbf{T} - \frac{1}{J^e}\mathbf{F}^e\mathbf{T}^p(\mathbf{F}^e)^\top, \tag{8.33}$$

where the deviatoric back stress is given by the incompressible eight-chain model which can be written:

$$\mathbf{T}^p = \frac{\mu^p}{\overline{\lambda^p}} \frac{\mathcal{L}^{-1}\left(\overline{\lambda^p}/\lambda^p_{\mathrm{lock}}\right)}{\mathcal{L}^{-1}\left(1/\lambda^p_{\mathrm{lock}}\right)} \mathrm{dev}[\mathbf{b}^p] \tag{8.34}$$

with μ^p, $\lambda^p_{\mathrm{lock}}$ being physically motivated material constants, $\mathbf{b}^p = \mathbf{F}^p(\mathbf{F}^p)^\top$, $\overline{\lambda^p} = (\mathrm{tr}[\mathbf{b}^p]/3)^{1/2}$ the effective chain stretch based on the eight-chain topology assumption, and $\mathcal{L}^{-1}(x)$ the inverse Langevin function.

In the original work [16, 20, 21] the plastic flow rate was given by:

$$\dot{\gamma}^p = \dot{\gamma}_0 \exp\left[-\frac{As}{k_{\mathrm{B}}\theta}\left(1 - \left(\frac{\tau}{s}\right)^{5/6}\right)\right], \tag{8.35}$$

where $\dot{\gamma}_0$, A, s are material constants, k_{B} is Boltzmann's constant, and θ is the absolute temperature. It has been shown by Hasan and Boyce [23] that the difference in behavior between a stress exponent of 5/6 and 1 is very small. By taking the stress exponent to be 1 and grouping material constants together the expression for the plastic flow rate can be simplified to

$$\dot{\gamma}^p = \dot{\gamma}_i \exp\left[\frac{\tau}{\tau_{\mathrm{base}}}\right], \tag{8.36}$$

where $\dot{\gamma}_i$ and τ_{base} are material parameters. The focus of the current work is on isothermal deformation histories, to explicitly include temperature effects the parameter τ_{base} can be replaced by $k_{\mathrm{B}}\theta/A$. The scalar equivalent stress τ is here taken as the Frobenius norm of the deviatoric part of the driving stress

$\tau = ||\,\mathrm{dev}[\mathbf{T}^*]||_F$, where $||\mathbf{A}||_F \equiv (A_{ij}A_{ij})^{1/2}$. The rate of plastic deformation is given by

$$\mathbf{D}^P = \frac{\dot{\gamma}^p}{\tau}\,\mathrm{dev}[\mathbf{T}^*] \tag{8.37}$$

and the plastic spin is taken to be zero [16], i.e. $\mathbf{W}^p = 0$, which uniquely specifies the rate kinematics. The time-derivative of $\dot{\mathbf{F}}$ is given by

$$\dot{\mathbf{F}}^p = \mathbf{D}^p\mathbf{F}^p = \frac{\dot{\gamma}^p}{\tau}\,\mathrm{dev}[\mathbf{T}^*]\mathbf{F}^p. \tag{8.38}$$

Note that the original Boyce model also allows for modeling of strain softening through an evolution equation of the athermal shear resistance, s:

$$\dot{s} = h\left[1 - \frac{s}{s_{\mathrm{ss}}}\right]\dot{\gamma}^p, \tag{8.39}$$

where h and s_{ss} are material parameters. This equation can also be written in terms of the effective shear strength:

$$\dot{\tau}_{\mathrm{base}} = h\left[1 - \frac{\tau_{\mathrm{base}}}{\tau_{\mathrm{base.ss}}}\right]\dot{\gamma}^p, \tag{8.40}$$

where $\tau_{\mathrm{base.0}}$ is the initial flow resistance, and $\tau_{\mathrm{base.ss}}$ is the steady-state flow resistance at large strains.

Figure 8.17 shows one exemplar set of predictions from the AB-model. In this case, the response in uniaxial tension followed by unloading is shown at three different strain rates. In this example the flow resistance evolves from an initial value of 20 MPa down to 10 MPa at steady state (Figure 8.18).

In summary, the AB model is based on a simple theoretical framework, and as a result is easy to use but not always as accurate as needed for analysis of engineering problems.

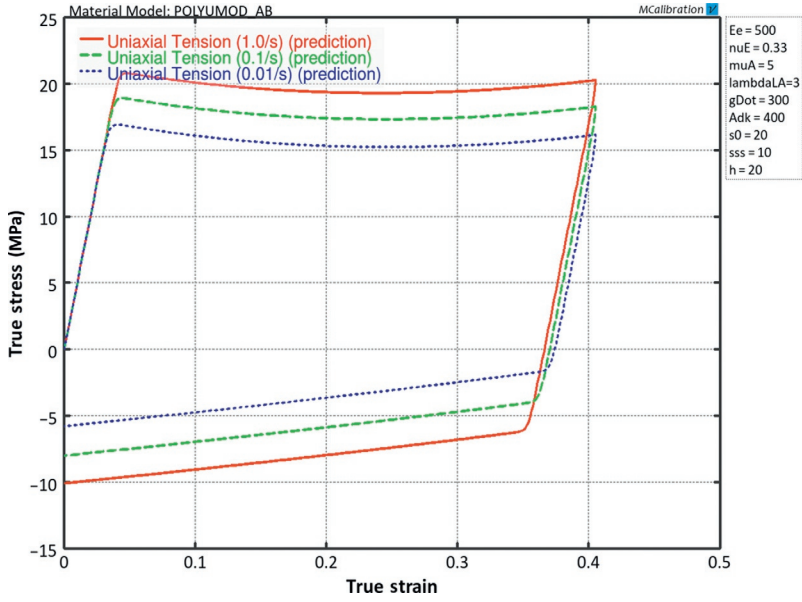


Figure 8.18 Stress-strain predictions from the Arruda-Boyce model.

8.4 Dual Network Fluoropolymer Model

The dual network fluoropolymer (DNF) model is an advanced material model capable of predicting the large-strain, time- and temperature-dependent viscoplastic behavior of various types of fluoropolymers and other types of thermoplastics.[3]

Fluoropolymers, as well as other thermoplastics, exhibit a complicated non-linear response when subjected to external loads. At small applied strains, the material response is typically linear viscoelastic. At larger strains, the material undergoes distributed yielding that evolves with plastic strain, followed by large-scale viscoplastic flow, and finally, gradual material stiffening at large strain until ultimate failure occurs. It is also known that the material response is strongly dependent on applied strain-rate and temperature: higher strain rates and lower temperatures increase the stiffness of the material. The DNF model is a constitutive model aimed at predicting these experimentally observed char-

[3] *The DNF model is available in the PolyUMod library [22].*

acteristics. The proposed model is an extension of previous work by Bergström and Boyce [1, 4, 5] and Arruda and Boyce [21] for elastomers and glassy polymers.

There are a number of different candidate material models that are documented in the literature for predicting the behavior of general thermoplastics (e.g. [21, 24–26]), and some of these models are summarized in this chapter. Development of advanced constitutive models for polymers is an active area of research that is evolving and improving. Up until about year 2000, there were no constitutive models specifically developed for fluoropolymers, and the most useful models were either classical isotropic plasticity, linear viscoelasticity models, or general models for thermoplastics [4, 21]. Since then constitutive models specifically developed for fluoropolymers have emerged [27, 28]. These models are typically better at predicting the experimentally observed characteristics of fluoropolymers than traditional isotropic plasticity or viscoelasticity models, but have limitations of isothermal conditions only.

The DNF model can be represented using the rheological model shown in Figure 8.19.

The DNF model incorporates experimental characteristics by using a decomposition of the material behavior into a viscoplastic response, corresponding to irreversible molecular chain sliding (due to the lack of chemical crosslinks in the material) and a time-dependent viscoelastic response. The viscoelastic response is further decomposed into the response of two molecular networks acting in parallel: a first network (A) captures the equilibrium (long term) of the viscoelastic response and a second network (B) the time-dependent (short term) deviation from the viscoelastic equilibrium state. A schematic illustrating the kinematics of deformation are shown in Figure 8.20.

The total deformation gradient $\mathbf{F}^{\text{appl}}$ contains both a thermal expansion part $\mathbf{F}^{\text{th}} = [1 + \alpha(\theta - \theta_0)]\,\mathbf{I}$, and a mechanical deformation part $\mathbf{F}$:

$$\mathbf{F}^{\text{appl}} = \mathbf{F}\,\mathbf{F}^{\text{th}}.$$

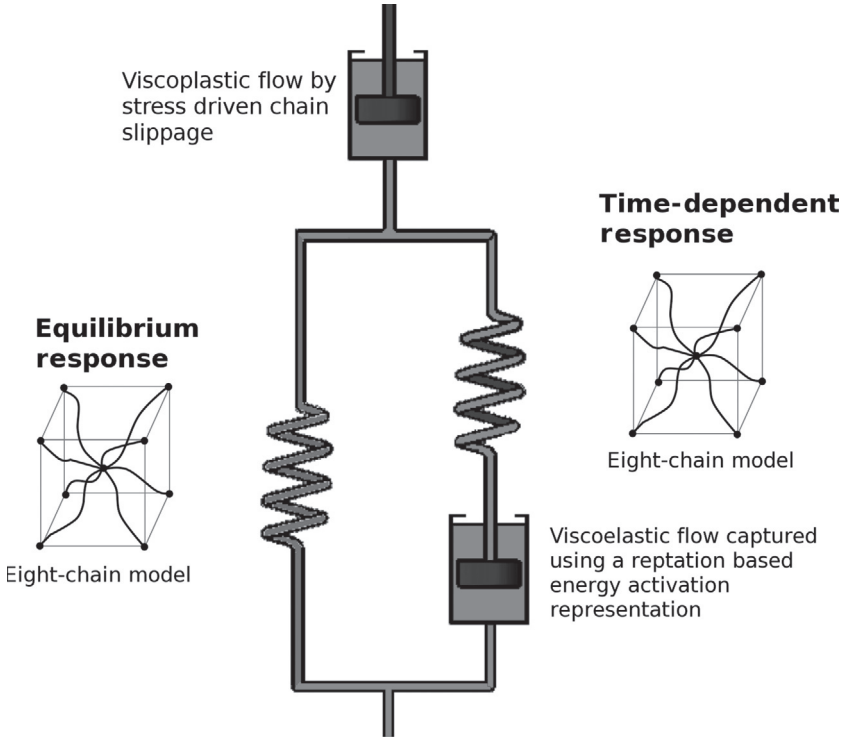


Figure 8.19 Structure of the dual network fluoropolymer model.

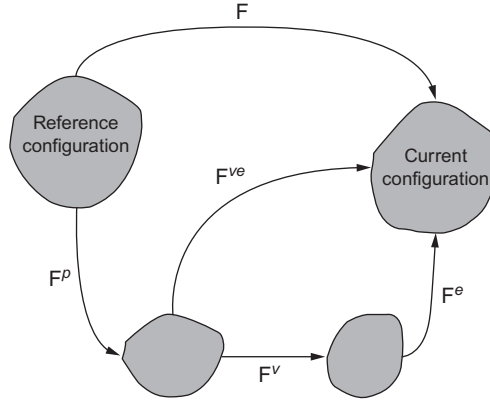


Figure 8.20 Kinematics of deformation for the DNF model.

The deformation gradient $\mathbf{F}$ is multiplicatively decomposed into viscoplastic and viscoelastic components:

$$\mathbf{F} = \mathbf{F}^{ve}\mathbf{F}^{p}. \tag{8.41}$$

The Cauchy stress acting on network A is given by the eight-chain representation [4, 19]:

$$\boldsymbol{\sigma}^{ve} = \mathbf{f}_{8ch}(\mathbf{F}^{ve}) = \frac{\mu_A^0(\theta)}{J^{ve}\overline{\lambda^*}} \frac{\mathcal{L}^{-1}\left(\overline{\lambda^{ve*}}/\lambda^{\text{lock}}\right)}{\mathcal{L}^{-1}\left(1/\lambda^{\text{lock}}\right)} \operatorname{dev}\left[\mathbf{B}^{ve*}\right] + \frac{\kappa \ln[J^{ve}]}{J^{ve}}\mathbf{1}, \tag{8.42}$$

where $J^{ve} = \det[\mathbf{F}^{ve}]$, $\mu_A^0(\theta)$ is a temperature-dependent initial shear modulus, λ^{lock} is the chain locking stretch, $\mathbf{b}^{ve*} = (J^{ve})^{-2/3}\mathbf{F}^{ve}(\mathbf{F}^{ve})^T$ is the Cauchy-Green deformation tensor, $\overline{\lambda^{ve*}} = (\text{tr}[\mathbf{b}^{ve*}]/3)^{1/2}$ is the effective chain stretch based on the eight-chain topology assumption [19], $\mathcal{L}^{-1}(x)$ is the inverse Langevin function, where $\mathcal{L}(x) = \coth(x) - 1/x$, and κ is the bulk modulus. By explicitly incorporating the temperature-dependence of the shear modulus it is possible to capture the stiffness variation of the material over a range of temperatures. The following expression is found to accurately capture the experimentally observed temperature dependence of the shear modulus for temperatures between 20 °C and 200 °C:

$$\mu_A(\theta) = \mu_A^0 \exp\left[\frac{\theta_0 - \theta}{\theta_{\text{base}}}\right], \tag{8.43}$$

where θ is the current temperature, and μ_A^0, θ_0, and θ_{base} are material parameters.

The viscoelastic deformation gradient acting on network B is decomposed into elastic and viscous parts:

$$\mathbf{F}^{ve} = \mathbf{F}^{e}\mathbf{F}^{v}. \tag{8.44}$$

The Cauchy stress acting on network B is obtained from the same eight-chain network representation that was used for network A. For simplicity, the response of network B is taken as a scalar factor s_B (a specified material parameter) times the eight-chain ex-

pression that was used for network A applied on the deformation gradient $\mathbf{F}^e$:

$$\boldsymbol{\sigma}^e = s_B \cdot \mathbf{f}_{8\text{ch}}(\mathbf{F}^e), \tag{8.45}$$

where $\mathbf{f}_{8\text{ch}}(\cdot)$ is the tensorial function defined in Equation (8.42). Using this framework, the total Cauchy stress in the system is given by $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{ve} + \boldsymbol{\sigma}^e$.

The total velocity gradient of network B, $\mathbf{L}^{ve} = \dot{\mathbf{F}}^{ve}(\mathbf{F}^{ve})^{-1}$, can similarly be decomposed into elastic and viscous components: $\mathbf{L}^{ve} = \mathbf{L}^e + \mathbf{F}^e\mathbf{L}^e\mathbf{F}^{e-1} = \mathbf{L}^e + \tilde{\mathbf{L}}^v$, where $\mathbf{L}^v = \dot{\mathbf{F}}^v\mathbf{F}^{v-1} = \mathbf{D}^v + \mathbf{W}^v$ and $\tilde{\mathbf{L}}^v = \tilde{\mathbf{D}}^v + \tilde{\mathbf{W}}^v$. The unloading process relating the deformed state with the intermediate state is not uniquely defined, since an arbitrary rigid body rotation of the intermediate state still leaves the state stress free. The intermediate state can be made unique in different ways [16], one particularly convenient way that is used here is to prescribe $\tilde{\mathbf{W}}^v = \mathbf{0}$. This will, in general, result in elastic and inelastic deformation gradients both containing rotations.

The rate of viscoplastic flow of network B is constitutively prescribed by

$$\tilde{\mathbf{D}}^v = \dot{\gamma}^v_{\text{dev}}\mathbf{N}^v_{\text{dev}} + \dot{\gamma}^v_{\text{vol}}\mathbf{N}^v_{\text{vol}}, \tag{8.46}$$

where the first term gives the *deviatoric* viscoelastic flow and the second term gives the *volumetric* viscoelastic flow. The tensors $\mathbf{N}^v_{\text{dev}}$ and $\mathbf{N}^v_{\text{vol}}$ specify the directions of the driving deviatoric and volumetric stresses of the relaxed configuration convected to the current configuration, and the terms $\dot{\gamma}^v_{\text{dev}}$ and $\dot{\gamma}^v_{\text{vol}}$ specify the effective deviatoric and volumetric flow rates. Noting that $\boldsymbol{\sigma}^e$ is computed in the loaded configuration, the driving deviatoric stress on the relaxed configuration convected to the current configuration is given by $\boldsymbol{\sigma}^{e\prime} = \text{dev}[\boldsymbol{\sigma}^e]$, and by defining an effective stress by the Frobenius norm $\tau^e = ||\boldsymbol{\sigma}^{e\prime}||_F \equiv \left(\text{tr}[\boldsymbol{\sigma}^{e\prime}\boldsymbol{\sigma}^{e\prime}]\right)^{1/2}$, the direction of the driving deviatoric stress becomes $\mathbf{N}^v_{\text{dev}} = \boldsymbol{\sigma}^{e\prime}/\tau^e$. The effective deviatoric flow rate is given by the reptation-inspired equation [4]:

$$\dot{\gamma}^v_{\text{dev}} = \dot{\gamma}_0\left[\overline{\lambda^v} - 1 + \xi\right]^C \cdot \left(\frac{\tau^e}{\tau_{\text{base}} + \beta R(p^e)}\right)^m \cdot \left(\frac{\theta}{\theta_{\text{base}}}\right)^n, \tag{8.47}$$

where $\overline{\lambda^v} = \sqrt{\text{tr}[\mathbf{B}^{v*}]/3}$ is an effective viscoelastic chain stretch, $\mathbf{B}^{v*} = (J^v)^{-2/3}\mathbf{F}^v(\mathbf{F}^v)^T$ is the Cauchy-Green deformation tension, $R(\cdot)$ is the ramp function, $p^e = -(\boldsymbol{\sigma}^e_{11} + \boldsymbol{\sigma}^e_{22} + \boldsymbol{\sigma}^e_{33})/3$ is the hydrostatic pressure, $\dot{\gamma}_0$ is a constant taken as 1/s (1 reciprocal second) that is needed for dimensional consistency, and C, β, m, τ_{base}, n, and θ_{base} are specified material parameters.

In this framework, the temperature dependence of the flow rate is taken to follow a power law form. Due to the high bulk modulus of PTFE the effective volumetric flow rate is small and is here simply represented with a constant viscosity η_{vol}:

$$\dot{\gamma}^v_{\text{vol}} = -p^e/\eta_{\text{vol}}. \tag{8.48}$$

In summary, the velocity gradient of the viscoelastic flow can be written

$$\dot{\mathbf{F}}^v = \mathbf{F}^{e-1}\left(\dot{\gamma}^v_{\text{dev}}\frac{\text{dev}[\boldsymbol{\sigma}^e]}{\tau^e} + \dot{\gamma}^v_{\text{vol}}\mathbf{I}\right)\mathbf{F}^{ve}. \tag{8.49}$$

The rate of plastic flow is captured by a simple phenomenological representation:

$$\dot{\gamma}^p = \begin{cases} ab(\epsilon - \epsilon_0)^{b-1}\dot{\epsilon} & \text{if } \tau > \sigma_0, \\ 0 & \text{otherwise,} \end{cases} \tag{8.50}$$

where $a > 0$, $b > 0$ and $\sigma_0 > 0$ are material parameters, $\tau = ||\,\text{dev}[\boldsymbol{\sigma}]||_F$ is the Frobenius norm of the deviatoric portion of the Cauchy stress $\boldsymbol{\sigma}$, and ϵ_0 is the effective strain at which τ becomes equal to σ_0. The effective strain in Equation (8.50) is obtained from $\epsilon = ||\mathbf{E}_{\ln}||_F$, where $\mathbf{E}_{\ln} = \ln[\mathbf{V}]$ is the logarithmic strain, and $\dot{\epsilon}$ is the rate of change of the effective strain. The key feature of Equation (8.50) is that it predicts the rate of plastic flow to be proportional to the applied strain rate and the magnitude of the current strain. By inserting $\mathbf{F} = \mathbf{F}^{ve}\mathbf{F}^p$ into $\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1}$, the expression for the velocity gradient can be expanded to

$\mathbf{L} = \mathbf{L}^{ve} + \mathbf{F}^{ve}\mathbf{L}^{p}\mathbf{F}^{ve} = \mathbf{L}^{ve}\tilde{\mathbf{L}}^{p}$. By taking $\tilde{\mathbf{W}}^{p} = \mathbf{0}$, the viscoplastic velocity gradient can be written $\tilde{\mathbf{L}}^{p} = \dot{\gamma}^{p}\,\mathrm{dev}[\boldsymbol{\sigma}]/\tau$ giving

$$\dot{\mathbf{F}}^{p} = \dot{\gamma}^{p}\mathbf{F}^{ve-1}\frac{\mathrm{dev}[\boldsymbol{\sigma}]}{\tau}\mathbf{F}, \tag{8.51}$$

specifying the rate kinematics of the plastic flow.

The DNF model requires the material parameters in Table 8.1.

Table 8.1 Material Parameters Used by the DNF Model

Index	Symbol	Description
1	μ_A^0	Shear modulus of network A
2	θ_0	Reference temperature
3	θ_{base}	Temperature factor
4	λ^{lock}	Locking stretch
5	κ	Bulk modulus
6	s_B	Relative stiffness of network B
7	ξ	Strain adjustment factor
8	C	Strain exponential
9	τ_{base}	Flow resistance
10	β	Pressure dependence of flow
11	m	Stress exponential
12	n	Temperature exponential
13	η_{vol}	Volumetric flow viscosity
14	a	Plastic flow ratio
15	b	Plastic flow exponent
16	σ_0	Plastic flow strength
17	α	Thermal expansion coefficient

8.4.1 Matlab Implementation of the DNF Model

The DNF material model is formulated as a set of differential equations that need to be solved for each time increment. One way to solve this set of equations is to use the following algorithm:

1. Known values at time t_i:
   - Deformation gradient: $\mathbf{F}$
   - State variables: $\mathbf{F}^v$, $\mathbf{F}^p$
2. Known values at time t_{i+1}:
   - Deformation gradient: $\mathbf{F}$
3. Calculate $\mathbf{F}^v$ at t_{i+1} using an ODE solver and Equation (8.49) for $\dot{\mathbf{F}}^v$.
4. Calculate $\mathbf{F}^p$ at t_{i+1} using an ODE solver and Equation (8.51) for $\dot{\mathbf{F}}^p$.
5. Calculate $\boldsymbol{\sigma}$ using Equation (8.42), and Equation (8.45) at t_{i+1}.

For the case of incompressible, uniaxial loading this algorithm can be implemented into Matlab code as shown in Figure 8.21.

Example code that can be used to calculate the stress-strain response is shown in Figure 8.22. The results from running this code are shown in Figure 8.23.

8.4.2 Use of the DNF Model for Polymer Modeling

One example of how the DNF model can be used to predict the response of a fluoropolymer with 15 vol% short glass fibers is shown in this section. The material model calibrated to the complete set of uniaxial tension and compression data, and all predictions shown in Figures 8.24–8.28 were obtained using the same set of material parameters. As is shown in these figures, the DNF model can accurately predict the characteristic material response.

Matlab File Name: mat_DNF.m

```
function [stress] = mat_DNF(time, strain, params)
%mat_DNF Dual-Network Fluoropolymer Model
%Incompressible uniaxial loading (uses true strain & stress)
%Parameters: [muA lambdaL sB xi C tauBas beta m a b sigma0]
%State variabes: [lambdaBv lambdaP]
muA = params(1); lambdaL = params(2); sB = params(3);
stress = 0 * strain;
state = [1 1]';
for i = 2 : length(strain)
    [t,stateAll] = ode45(@(t,y) flow(t, y, time, strain, params), time(i-1:i), state);
    state = stateAll(end,:);
    lambda = exp(strain(i));
    lambdaB = lambda / state(2);
    lambdaBe = lambdaB / state(1);
    stressA = mat_EC(time(i), log(lambdaB), [muA lambdaL]);
    stressB = mat_EC(time(i), log(lambdaBe), [sB*muA lambdaL]);
    stress(i) = stressA + stressB;
end
end

function res = flow(time, state, timeVec, strainVec, params)
% Calculates the time-derivative of the state variables
global eps0
muA     = params( 1); lambdaL = params( 2); sB      = params( 3);
xi      = params( 4); C       = params( 5); tauBas  = params( 6);
beta    = params( 7); m       = params( 8); a       = params( 9);
b       = params(10); sigma0  = params(11);
lambda = exp( interp1(timeVec, strainVec, time) ); % stretch at the specified time
lambdaBv = state(1);
lambdaP = state(2);
lambdaB = lambda / state(2);
lambdaBe = lambdaB / lambdaBv;
% network B
lambdaBvChain = sqrt((lambdaBv^2 + 2 / lambdaBv) / 3);
stressB = mat_EC(time, log(lambdaBe), [sB*muA lambdaL]);
gammaBDot = (lambdaBvChain-1+xi)^C * (abs(stressB) / (tauBas+beta*stressB))^m;
res(1,1) = gammaBDot * sign(stressB) * lambdaBv;
% network P
strain = log(lambda);
dt = 1e-4; % simple estimation of the strain rate
strain1 = interp1(timeVec, strainVec, time+dt, 'linear', 'extrap');
strainDot = (strain1 - strain) / dt;
stressA = mat_EC(time, log(lambdaB), [muA lambdaL]);
stress = stressA + stressB;
if abs(stress) > sigma0
    if eps0 == 0
        eps0 = strain;
        res(2,1) = 0;
    else
        gamDotP = a*b*abs(strain - eps0)^(b-1) * strainDot;
        res(2,1) = gamDotP * sign(stress) * lambdaP;
    end
else
    eps0 = 0;
    res(2,1) = 0;
end
end
```

Figure 8.21 Matlab implementation of the DNF Model.

```
% Create the Dual Network Fluoropolymer Model prediction
time = linspace(0,2)';
strain = [linspace(0, 0.5,50)'; linspace(0.5, 0, 50)'];

muA = 10;
lambdaL = 3;
sB = 4;
xi = 0.05;
C = -0.5;
tauBas = 20;
beta = 0.00;
m = 4;
a = 0.02;
b = 1.10;
sigma0 = 10;
params = [muA lambdaL sB xi C tauBas beta m a b sigma0];

stress = mat_DNF(time, strain, params);
```

Figure 8.22 Matlab test code for the DNF Model.

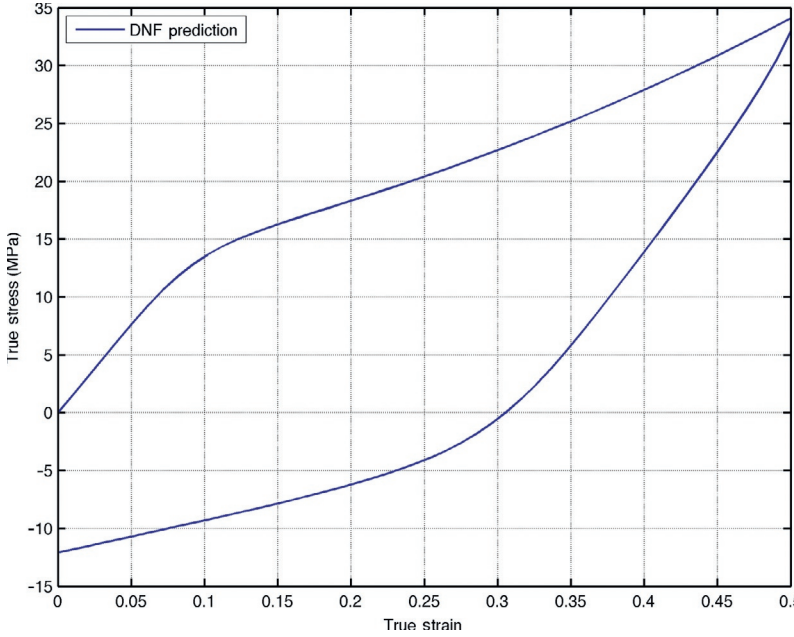


Figure 8.23 Predicted stress-strain results from the Matlab implementation of the DNF model.

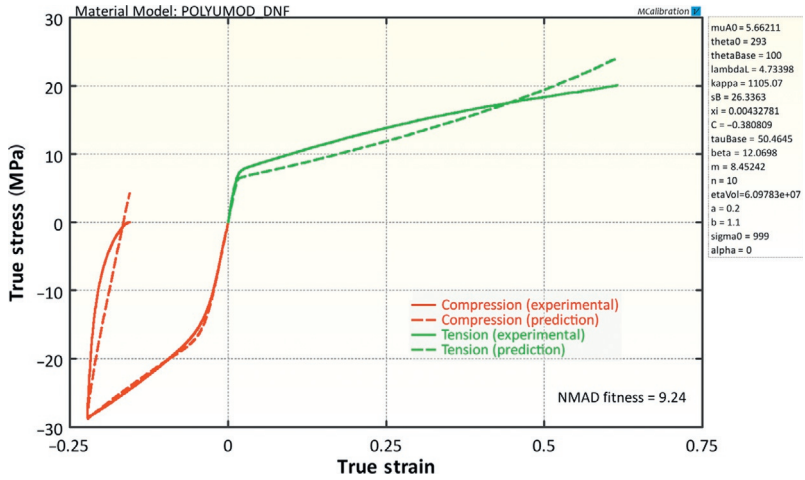


Figure 8.24 Comparison between experimental data in uniaxial tension and compression and model predictions from the DNF model.

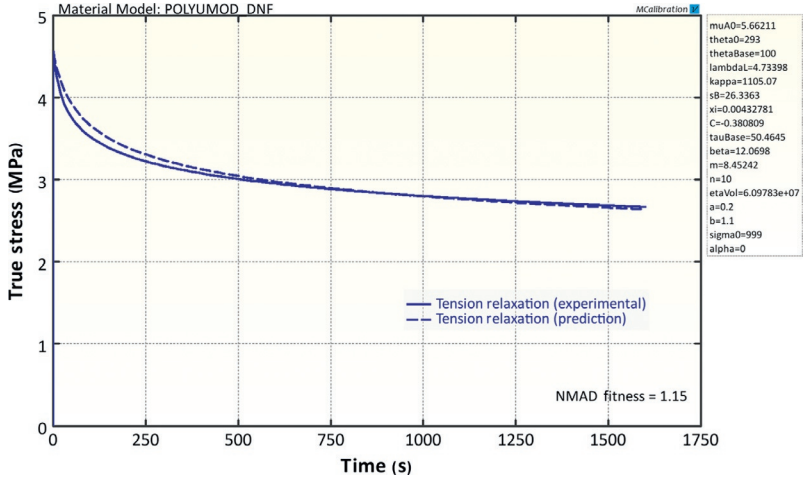


Figure 8.25 Comparison between experimental stress relaxation data and model predictions from the DNF model.

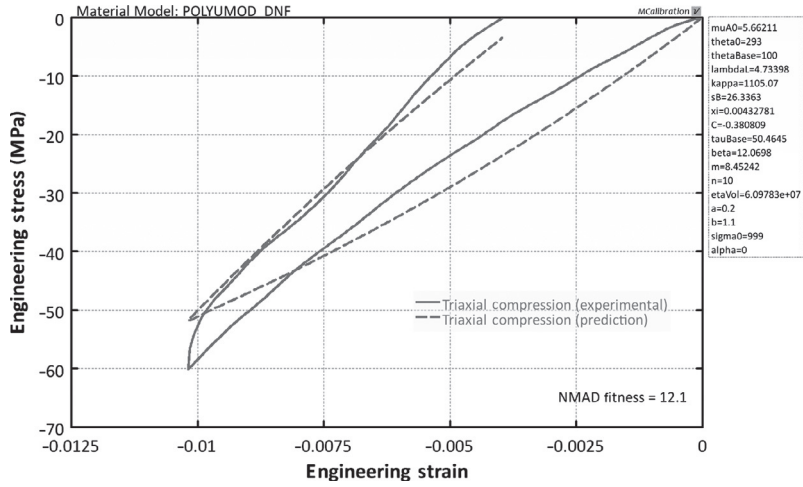


Figure 8.26 Comparison between experimental volumetric compression data and model predictions from the DNF model.

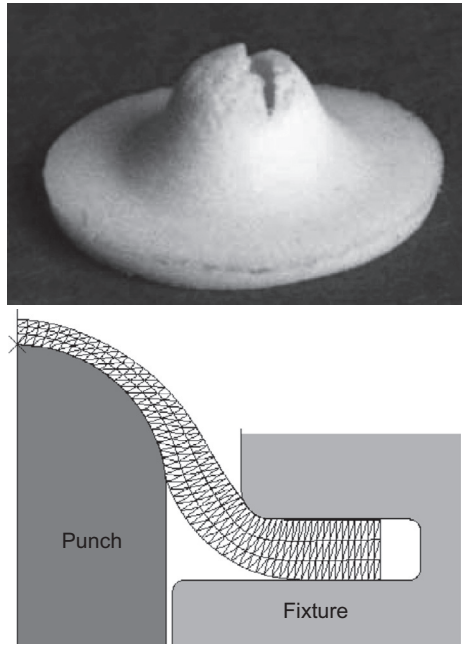


Figure 8.27 Deformed specimen used in the punch experiments.

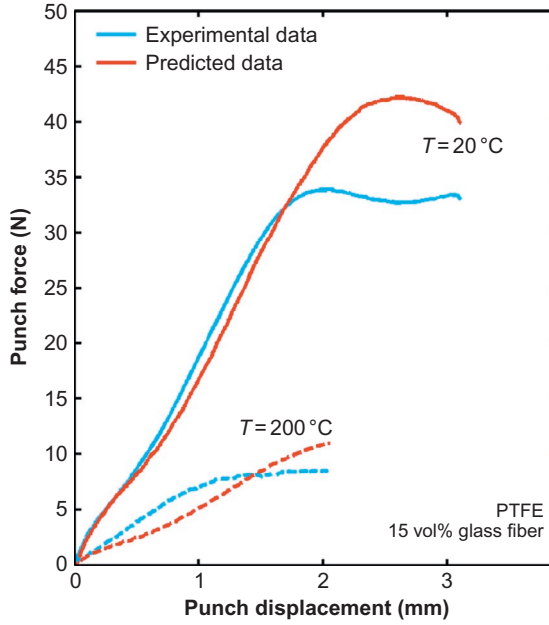


Figure 8.28 Comparison between experimental data for a fluoropolymer and model predictions from the DNF model.

8.5 Hybrid Model

The Hybrid Model (HM) is an advanced material model specifically developed for predicting the large strain time-dependent behavior of ultra-high molecular weight polyethylene (UHMWPE) [24, 29]. This model can also be used to predict the response of many other types of thermoplastics.[4]

The kinematic framework used in the HM is based on a decomposition of the applied deformation gradient into elastic and viscoplastic components: $\mathbf{F} = \mathbf{F}^e\mathbf{F}^p$, see Figure 8.29.

The model can also be represented using the rheological representation shown in Figure 8.30.

[4] *The HM model is available in the PolyUMod library [22].*

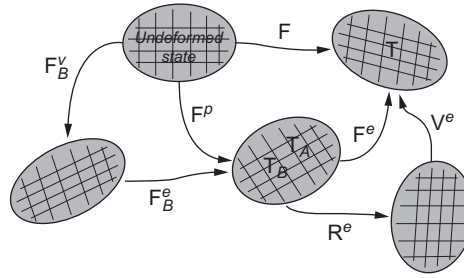


Figure 8.29 Deformation map used in the Hybrid Model.

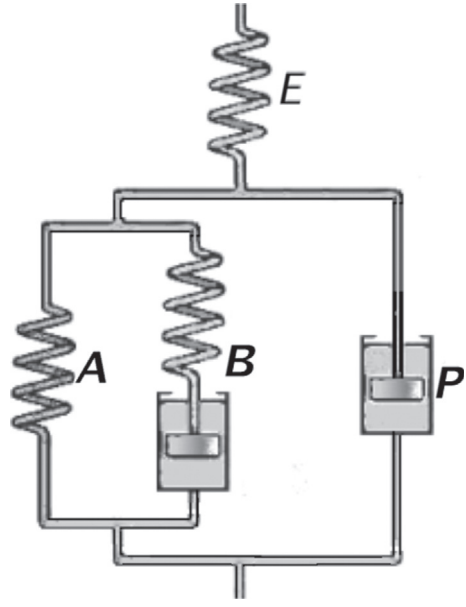


Figure 8.30 Rheological representation of the Hybrid Model.

The Cauchy stress for the HM at a given deformation state is given by the isotropic linear elasticity expression:

$$\boldsymbol{\sigma} = \frac{1}{J^e}\left(2\mu\mathbf{E}^e + \lambda\,\mathrm{tr}[\mathbf{E}^e]\mathbf{I}\right), \tag{8.52}$$

where $\mathbf{V}^e$ is left stretch tensor, $\mathbf{E}^e = \ln[\mathbf{V}^e]$ is the logarithmic strain, $J^e = \det[\mathbf{F}^e]$ is the relative volume change, and μ and λ are the Lamé constants that can be obtained from the Young's modulus (E) and the Poisson's ratio (ν) from:

$$\mu = \frac{E}{2(1+\nu)},$$
$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}.$$

The stress acting on the equilibrium portion of the backstress network A is given by the eight-chain model (see Section 5.3.10):

$$\boldsymbol{\sigma}_A = \frac{\mu_A}{J^p\overline{\lambda^{p*}}}\frac{\mathcal{L}^{-1}\left(\overline{\lambda^{p*}}/\lambda^{\text{lock}}\right)}{\mathcal{L}^{-1}\left(1/\lambda^{\text{lock}}\right)}\,\text{dev}\left[\mathbf{b}^{p*}\right] + \kappa(J^p-1)\mathbf{I}, \quad (8.53)$$

where $J^p = \det[\mathbf{F}^p]$, $\mathbf{b}^{p*} = (J^p)^{-2/3}\mathbf{F}\mathbf{F}^\top$ is the distortional part of the left Cauchy-Green deformation tensor, $\overline{\lambda^{p*}} = (\text{tr}[\mathbf{b}^{p*}]/3)^{1/2}$ is the chain stretch in network A. This hyperelastic network requires three material parameters: a shear modulus μ_A, a locking stretch λ^{lock}, and a bulk modulus κ.

The kinematics of the viscoelastic flow portion of the backstress network B is represented by an approach similar to the BB model (see Section 8.2). Specifically, the deformation gradient is decomposed into elastic and viscoelastic components: $\mathbf{F}^p = \mathbf{F}^e_B\mathbf{F}^v_B$. The stress driving the viscoplastic flow of the backstress network is obtained from the eight-chain model:

$$\boldsymbol{\sigma}_B = s_B\frac{\mu_A}{J^e_B\overline{\lambda^{e*}_B}}\frac{\mathcal{L}^{-1}\left(\overline{\lambda^{e*}_B}/\lambda^{\text{lock}}\right)}{\mathcal{L}^{-1}\left(1/\lambda^{\text{lock}}\right)}\,\text{dev}\left[\mathbf{b}^{e*}_B\right] + \kappa(J^e_B-1)\mathbf{I}, \quad (8.54)$$

where $J^e_B = \det[\mathbf{F}^e_B]$, $\mathbf{b}^{e*}_B = (J^e_B)^{-2/3}\mathbf{F}^e_B(\mathbf{F}^e_B)^\top$, $\overline{\lambda^{e*}_B} = (\text{tr}[\mathbf{b}^{e*}_B]/3)^{1/2}$, and s_B is a dimensionless parameter specifying the relative stiffness of network B compared to network A.

At small deformations the stiffness of the backstress network is constant and the material response is linear elastic. At intermediate applied deformations viscoplastic flow is initiated by molecular chain sliding. With increasing amount of viscoplastic flow, the crystalline domains become distorted and start to provide additional molecular material to the backstress network. This is manifested by an initial reduction in the effective stiffness of the backstress network with imposed strain and is represented in the model by allowing the parameter s_B to evolve during the plastic deformation as follows

$$\dot{s}_B = -\alpha_B \cdot (s_B - s_{BF}) \cdot \dot{\gamma}_p, \tag{8.55}$$

where α_B is a material parameter specifying the transition rate of the distributed yielding, and s_B in the undeformed state is s_{Bi}, and s_{Bf} in the fully transformed state. The quantity $\dot{\gamma}_P$ is the rate of viscoplastic flow and is given by Equation (8.57).

The time derivative of the viscoelastic deformation gradient of network B is given by

$$\dot{\mathbf{F}}_B^v = \mathbf{L}_B^v \mathbf{F}_B^v = \dot{\gamma}_0 \cdot \left(\frac{\tau_B}{\tau_B^{\text{base}}[1 + p_B/\hat{p}]} \right)^{m_B}$$

$$(\mathbf{F}_B^e)^{-1} \frac{\text{dev}[\boldsymbol{\sigma}_B]}{\tau_B} \mathbf{F}^p \equiv \dot{\gamma}_B^v \mathbf{N}_B^e, \tag{8.56}$$

where $\dot{\gamma}_0 \equiv 1/\text{s}$ is a constant that is introduced to maintain dimensional consistency, $\tau_B = ||\,\text{dev}[\boldsymbol{\sigma}_B]||_F$ is the effective shear stress driving the viscoelastic flow, $p_B = -\text{tr}[\boldsymbol{\sigma}_B]/3$ is the hydrostatic pressure, and τ_B^{base}, m_B, and $\hat{p}$ are material parameters.

The time rate of change of the plastic flow of network P is captured using a similar energy activation approach as for network B

$$\dot{\mathbf{F}}^p = \mathbf{L}^p \mathbf{F}^p = \dot{\gamma}_0 \cdot \left(\frac{\tau^p}{\tau_{\text{base}}^p[1 + p^p/\hat{p}]} \right)^{m^p}$$

$$(\mathbf{R}^e)^\top \frac{\text{dev}[\mathbf{T}^p]}{\tau^p} \mathbf{R}^e \mathbf{F}^p \equiv \dot{\gamma}^p \mathbf{N}^p, \tag{8.57}$$

where $\dot{\gamma}_0 \equiv 1/\text{s}$ is a constant that is introduced to maintain dimensional consistency, $\tau^p = ||\,\text{dev}[\mathbf{T}^p]||_F$ is the effective shear stress driving the plastic flow, $p^p = -\text{tr}[\mathbf{T}^p]/3$ is the hydrostatic pressure, and τ_{base}^p, m^p, and $\hat{p}$ are material parameters.

In total, the augmented HM requires 13 material parameters (see Table 8.2): 2 small strain elastic constants (E_e, ν_e), 3 hyperelastic constants for the back stress network (μ_A, λ^{lock}, κ), 6 flow constants for the backstress network (s_{Bi}, s_{Bf}, α_B, τ_B^{base}, m_B, $\hat{p}$), and two yield and viscoplastic flow parameters (τ_{base}^p, m^p).

Table 8.2 Material Parameters Used by the Augmented Hybrid Model

Index	Symbol	Description
1	E	Young's modulus
2	ν	Poisson's ratio
3	μ_A	Shear modulus A
4	λ^{lock}	Locking stretch
5	κ	Bulk modulus
6	s_{Bi}	Initial stiffness B
7	s_{Bf}	Final stiffness B
8	α_B	Transition rate stiffness B
9	τ^B_{base}	Flow resistance B
10	m_B	Stress exponent B
11	$\hat{p}$	Pressure dependence of flow
12	τ^p_{base}	Flow resistance p
13	m^p	Stress exponent p

8.5.1 Matlab Implementation of the Hybrid Model

The HM material model is formulated as a set of differential equations that need to be solved for each time increment (Figure 8.31). One way to solve this set of equations is to use the following algorithm:

1. Known values at time t_i:
   - Deformation gradient: $\mathbf{F}$
   - State variables: $\mathbf{F}^v_B$, $\mathbf{F}^p$
2. Known values at time t_{i+1}:
   - Deformation gradient: $\mathbf{F}$
3. Calculate $\mathbf{F}^v_B$ at t_{i+1} using an ODE solver and Equation (8.56) for $\dot{\mathbf{F}}^v_B$.

Matlab File Name: mat_HM.m

```
function [stress] = mat_HM(time, strain, params)
%mat_HM Hybrid model
%Incompressible uniaxial loading (true stress & strain)
%Params: [E, muA, lambdaL, sBi, sBf, alphaB, tauBasB, mB, ...
%         pHat, tauBasP, mP]
%State variables: [lambdaBv, lambdaP, sB]
stress = 0 * strain;
state = [1 1 params(4)]';
for i = 2 : length(strain)
    [t,stateAll] = ode45(@(t,y) flow(t, y, time, strain, params), time(i-1:i), state);
    state = stateAll(end,:);
    stress(i) = params(1) * (strain(i) - log(state(2)));
end
end

function res = flow(time, state, timeVec, strainVec, params)
% Calculates the time-derivative of the state variables
E       = params( 1); % initialize variables
muA     = params( 2);
lambdaL = params( 3);
sBf     = params( 5);
alphaB  = params( 6);
tauBasB = params( 7);
mB      = params( 8);
pHat    = params( 9);
tauBasP = params(10);
mP      = params(11);
lambdaBv = state(1);
lambdaP  = state(2);
sB       = state(3);
% calculate stresses
lambda = exp( interp1(timeVec, strainVec, time) ); % stretch at the specified time
stress = E * log(lambda/lambdaP);
stressA = mat_EC(time, log(lambdaP), [muA lambdaL]);
stressB = mat_EC(time, log(lambdaP/lambdaBv), [sB*muA lambdaL]);
stressP = stress - (lambda/lambdaP)^2 * (stressA + stressB);
% calculate time-derivatives of state variables
res(1,1) = (abs(stressB) / (tauBasB * (1+stressB/pHat)))^mB * ...
    lambdaBv * sign(stressB); % lambdaBvDot
res(2,1) = (abs(stressP) / (tauBasP*(1+stressP/pHat)))^mP * ...
    lambdaP * sign(stressP); % lambdaPDot
res(3,1) = -alphaB * (sB - sBf) * ...
    (abs(stressP) / (tauBasP*(1+stressP/pHat)))^mP; % sBDot
end
```

Figure 8.31 Matlab implementation of the Hybrid Model.

4. Calculate $\mathbf{F}^p$ at t_{i+1} using an ODE solver and Equation (8.57) for $\dot{\mathbf{F}}^p$.
5. Calculate the stress at t_{i+1} using Equation (8.52).

For the case of incompressible uniaxial loading, the results from running this function are shown in Figure 8.32.

8.5.2 Use of the Hybrid Model for Polymer Modeling

The HM is useful for predicting the mechanical response of thermoplastic materials below the glass transition temperature, or semi-crystalline polymers below the melting temperature.

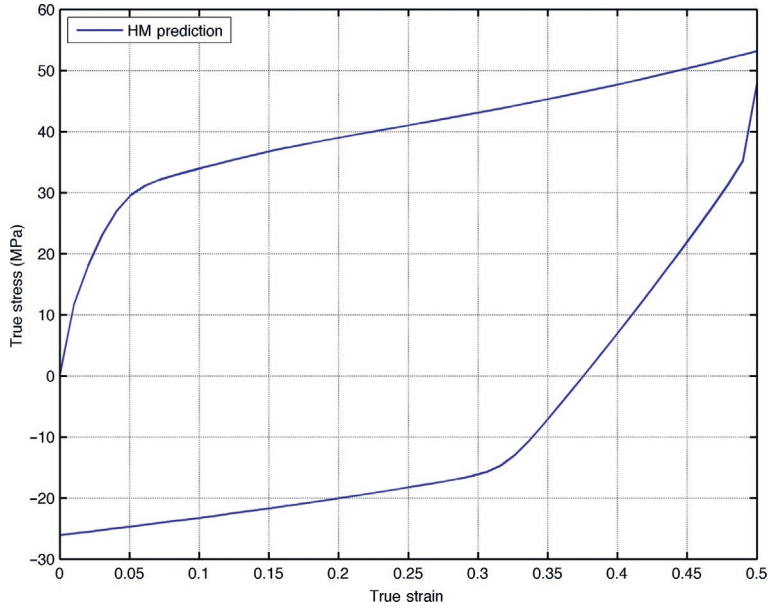


Figure 8.32 Matlab test of the Hybrid Model.

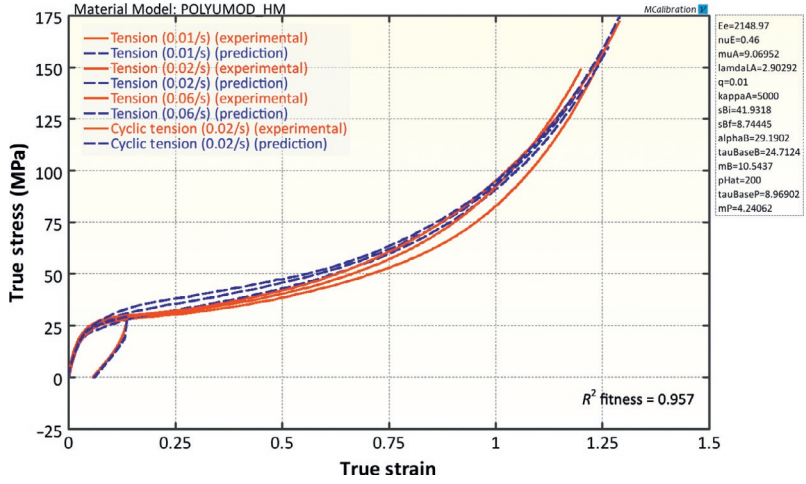


Figure 8.33 Comparison between experimental data in uniaxial tension for UHMWPE and predictions from the Hybrid Model.

The HM was originally developed for predicting the response of UHMWPE in biomedical applications [24, 29], but the model is suitable also for other thermoplastics.

Figures 8.33–8.35 compare the predictions from the HM to experimental data for UHMWPE (crosslinked GUR 1050). The

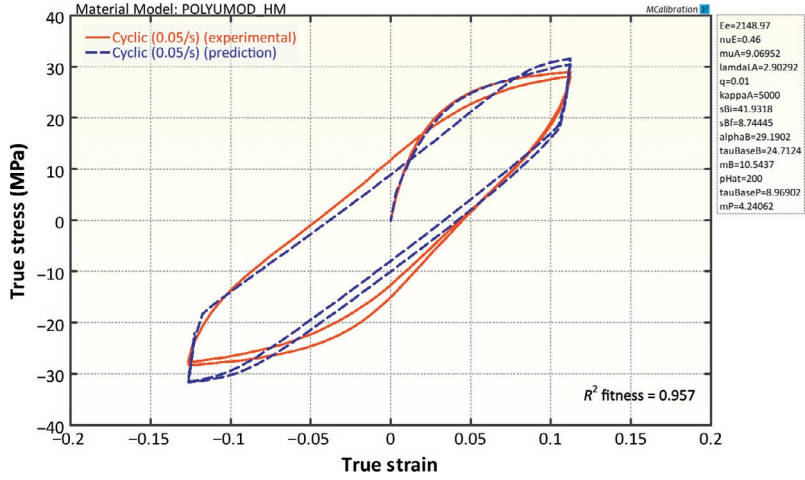


Figure 8.34 Comparison between cyclic experimental data for UHMWPE and predictions from the Hybrid Model.

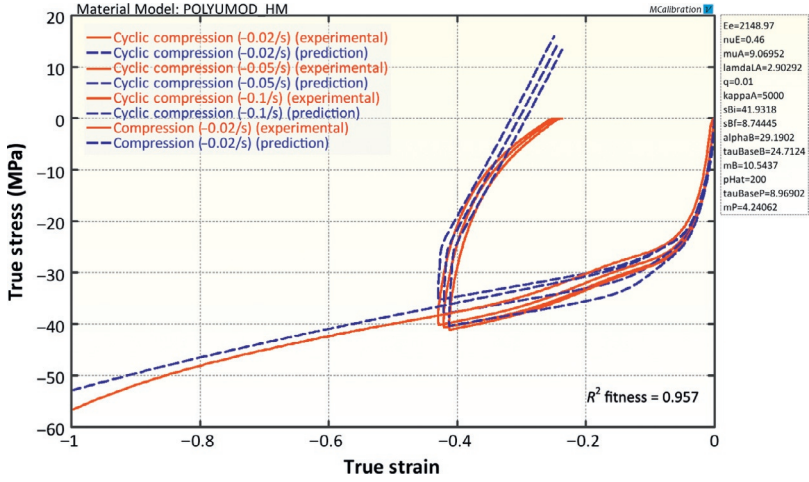


Figure 8.35 Comparison between experimental data in uniaxial compression for UHMWPE and predictions from the Hybrid Model.

material model was calibrated to all uniaxial tension and compression data simultaneously. The results are here shown in three separate images in order to better illustrate the predictive capabilities of the model. These images show that the HM can accurately predict the behavior of this material.

8.6 Three Network Model

The Three Network Model (TNM) is a material model specifically developed for thermoplastic materials. It has many features that are similar to the HM, but is designed to be more numerically efficient.[5]

As specified by its name, the kinematics of the TNM consists of three parts, or molecular networks, acting in parallel, see the rheological representation in Figure 8.36.

The total deformation gradient $\mathbf{F}^{\text{appl}}$ contains both a thermal expansion part $\mathbf{F}^{\text{th}} = [1 + \alpha(\theta - \theta_0)]\,\mathbf{I}$, and a mechanical deformation part $\mathbf{F}$:

$$\mathbf{F}^{\text{appl}} = \mathbf{F}\,\mathbf{F}^{\text{th}}.$$

The deformation gradient acting on network A is multiplicatively decomposed into viscoplastic and viscoelastic components:

$$\mathbf{F} = \mathbf{F}_A^e \mathbf{F}_A^v. \tag{8.58}$$

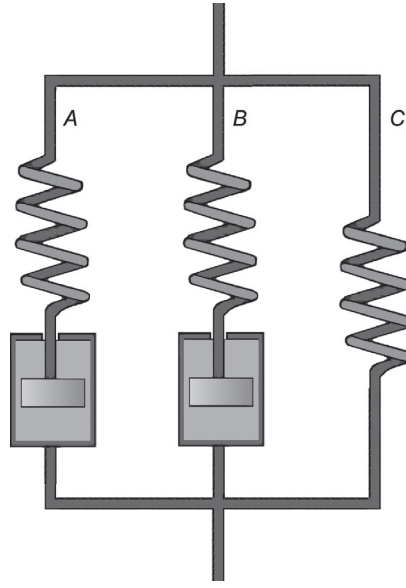


Figure 8.36 Rheological representation of the constitutive model.

[5] *The TN model is available in the PolyUMod library [22].*

The Cauchy stress acting on network A is given by the eight-chain representation [4, 19]:

$$\mathbf{T}_A = \frac{\mu_A}{J_A^e \overline{\lambda_A^{e*}}} \left[1 + \frac{\theta - \theta_0}{\hat{\theta}} \right] \frac{\mathcal{L}^{-1}\left(\frac{\overline{\lambda_A^{e*}}}{\lambda^{\text{lock}}} \right)}{\mathcal{L}^{-1}\left(\frac{1}{\lambda^{\text{lock}}} \right)} \text{dev}\left[\mathbf{b}_A^{e*} \right] + \kappa (J_A^e - 1)\mathbf{1}, \tag{8.59}$$

where $J_A^e = \det[\mathbf{F}_A^e]$, μ_A is the initial shear modulus, λ^{lock} is the chain locking stretch, θ is the current temperature, $\theta_0 = 293\,\text{K}$ is a fixed reference temperature, $\hat{\theta}$ is a material parameter specifying the temperature response of the stiffness, $\mathbf{b}_A^{e*} = (J_A^e)^{-2/3} \mathbf{F}_A^e (\mathbf{F}_A^e)^\top$ is a Cauchy-Green deformation tensor, $\overline{\lambda_A^{e*}} = \left(\text{tr}[\mathbf{b}_A^{e*}]/3 \right)^{1/2}$ is the effective chain stretch based on the eight-chain topology assumption [19], $\mathcal{L}^{-1}(x)$ is the inverse Langevin function, where $\mathcal{L}(x) = \coth(x) - 1/x$, and κ is the bulk modulus. By explicitly incorporating the temperature dependence of the shear modulus it is possible to capture the stiffness variation of the material over a wide range of temperatures.

The viscoelastic deformation gradient acting on network B is decomposed into elastic and viscous parts:

$$\mathbf{F} = \mathbf{F}_B^e \mathbf{F}_B^v. \tag{8.60}$$

The Cauchy stress acting on network B is obtained from the same eight-chain network representation that was used for network A.

$$\mathbf{T}_B = \frac{\mu_B}{J_B^e \overline{\lambda_B^{e*}}} \left[1 + \frac{\theta - \theta_0}{\hat{\theta}} \right] \frac{\mathcal{L}^{-1}\left(\frac{\overline{\lambda_B^{e*}}}{\lambda^{\text{lock}}} \right)}{\mathcal{L}^{-1}\left(\frac{1}{\lambda^{\text{lock}}} \right)} \text{dev}\left[\mathbf{b}_B^{e*} \right] + \kappa (J_B^e - 1)\mathbf{1}, \tag{8.61}$$

where $J_B^e = \det[\mathbf{F}_B^e]$, μ_B is the initial shear modulus, $\mathbf{b}_B^{e*} = (J_B^e)^{-2/3} \mathbf{F}_B^e (\mathbf{F}_B^e)^\top$ is a Cauchy-Green deformation tensor, and $\overline{\lambda_B^{e*}} = \left(\text{tr}[\mathbf{b}_B^{e*}]/3 \right)^{1/2}$ is the effective chain stretch based on the eight-chain topology assumption [19]. In Equation (8.61), the effective shear modulus is taken to evolve with plastic strain from an initial value of μ_{Bi} according to:

$$\dot{\mu}_B = -\beta \left[\mu_B - \mu_{Bf} \right] \cdot \dot{\gamma}_A, \tag{8.62}$$

where $\dot{\gamma}_A$ is the viscoplastic flow rate defined in Equation (8.64). This equation enables the model to better capture the distributed yielding that is observed in many thermoplastics.

Similarly the Cauchy stress acting on network C is given by

$$\mathbf{T}_C = \frac{\mu_C}{J\overline{\lambda^*}}\left[1 + \frac{\theta - \theta_0}{\hat{\theta}}\right]\frac{\mathcal{L}^{-1}\left(\frac{\overline{\lambda^*}}{\lambda^{\text{lock}}}\right)}{\mathcal{L}^{-1}\left(\frac{1}{\lambda^{\text{lock}}}\right)}\,\text{dev}\left[\mathbf{b}^*\right] + \kappa(J-1)\mathbf{1}, \tag{8.63}$$

where $J = \det[\mathbf{F}]$, μ_C is the initial shear modulus, $\mathbf{b}^* = J^{-2/3}\mathbf{F}(\mathbf{F})^\top$ is a Cauchy-Green deformation tensor, and $\overline{\lambda^*} = (\text{tr}[\mathbf{b}^*]/3)^{1/2}$ is the effective chain stretch based on the eight-chain topology assumption [19].

Using this framework, the total Cauchy stress in the system is given by $\mathbf{T} = \mathbf{T}_A + \mathbf{T}_B + \mathbf{T}_C$.

The total velocity gradient of network A, $\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1}$, can be decomposed into elastic and viscous components: $\mathbf{L} = \mathbf{L}_A^e + \mathbf{F}_A^e\mathbf{L}_A^v\mathbf{F}_A^{e-1} = \mathbf{L}_A^e + \tilde{\mathbf{L}}_A^v$, where $\mathbf{L}_A^v = \dot{\mathbf{F}}_A^v\mathbf{F}_A^{v-1} = \mathbf{D}_A^v + \mathbf{W}_A^v$ and $\tilde{\mathbf{L}}_A^v = \tilde{\mathbf{D}}_A^v + \tilde{\mathbf{W}}_A^v$. The unloading process relating the deformed state with the intermediate state is not uniquely defined since an arbitrary rigid body rotation of the intermediate state still leaves the state stress free. The intermediate state can be made unique in different ways [16], one particularly convenient way that is used here is to prescribe $\tilde{\mathbf{W}}_A^v = \mathbf{0}$. This will, in general, result in elastic and inelastic deformation gradients both containing rotations. The rate of viscoplastic flow of network A is constitutively prescribed by $\tilde{\mathbf{D}}_A^v = \dot{\gamma}_A\mathbf{N}_A$. The tensor $\mathbf{N}_A$ specifies the direction of the driving deviatoric stress of the relaxed configuration convected to the current configuration, and the term $\dot{\gamma}_A$ specifies the effective deviatoric flow rate. Noting that $\mathbf{T}_A$ is computed in the loaded configuration, the driving deviatoric stress on the relaxed configuration convected to the current configuration is given by $\mathbf{T}'_A = \text{dev}[\mathbf{T}_A]$, and by defining an effective stress by the Frobenius norm $\tau_A = ||\mathbf{T}'_A||_F \equiv \left(\text{tr}[\mathbf{T}'_A\mathbf{T}'_A]\right)^{1/2}$, the direction of the driving deviatoric stress becomes $\mathbf{N}_A = \mathbf{T}'_A/\tau_A$. The effective deviatoric flow rate is given by the reptation-inspired equation [4]:

$$\dot{\gamma}_A = \dot{\gamma}_0 \cdot \left(\frac{\tau_A}{\hat{\tau}_A + aR(p_A)} \right)^{m_A} \cdot \left(\frac{\theta}{\theta_0} \right)^n, \tag{8.64}$$

where $\dot{\gamma}_0 \equiv 1/\text{s}$ is a constant introduced for dimensional consistency, $p_A = -[(\mathbf{T}_A)_{11} + (\mathbf{T}_A)_{22} + (\mathbf{T}_A)_{33}]/3$ is the hydrostatic pressure, $R(x) = (x + |x|)/2$ is the ramp function, and $\hat{\tau}_A$, β, m_A, n, and θ_0 are specified material parameters. In this framework, the temperature dependence of the flow rate is taken to follow a power law form. In summary, the velocity gradient of the viscoelastic flow of network A can be written

$$\dot{\mathbf{F}}_A^v = \dot{\gamma}_A \mathbf{F}_A^{e-1} \frac{\text{dev}[\mathbf{T}_A]}{\tau_A} \mathbf{F}. \tag{8.65}$$

The total velocity gradient of network B can be obtained very similarly as for network A. Specifically, $\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1}$ can be decomposed into elastic and viscous components: $\mathbf{L} = \mathbf{L}_B^e + \mathbf{F}_B^e \mathbf{L}_B^v \mathbf{F}_B^{e-1} = \mathbf{L}_B^e + \tilde{\mathbf{L}}_B^v$, where $\mathbf{L}_B^v = \dot{\mathbf{F}}_B^v \mathbf{F}_B^{v-1} = \mathbf{D}_B^v + \mathbf{W}_B^v$ and $\tilde{\mathbf{L}}_B^v = \tilde{\mathbf{D}}_B^v + \tilde{\mathbf{W}}_B^v$. The unloading process relating the deformed state with the intermediate state is not uniquely defined since an arbitrary rigid body rotation of the intermediate state still leaves the state stress free. The intermediate state can be made unique in different ways [16], one particularly convenient way that is used here is to prescribe $\tilde{\mathbf{W}}_B^v = \mathbf{0}$. This will, in general, result in elastic and inelastic deformation gradients both containing rotations. The rate of viscoplastic flow of network B is constitutively prescribed by $\tilde{\mathbf{D}}_B^v = \dot{\gamma}_B \mathbf{N}_B$. The tensor $\mathbf{N}_B$ specifies the direction of the driving deviatoric stress of the relaxed configuration convected to the current configuration, and the term $\dot{\gamma}_B$ specifies the effective deviatoric flow rate. Noting that $\mathbf{T}_B$ is computed in the loaded configuration, the driving deviatoric stress on the relaxed configuration convected to the current configuration is given by $\mathbf{T}'_B = \text{dev}[\mathbf{T}_B]$, and by defining an effective stress by the Frobenius norm $\tau_B = ||\mathbf{T}'_B||_F \equiv \left(\text{tr}[\mathbf{T}'_B \mathbf{T}'_B]\right)^{1/2}$, the direction of the driving deviatoric stress becomes $\mathbf{N}_B = \mathbf{T}'_B/\tau_B$. The effective deviatoric flow rate is given by the reptation-inspired equation [4]:

$$\dot{\gamma}_B = \dot{\gamma}_0 \cdot \left(\frac{\tau_B}{\hat{\tau}_B + aR(p_B)} \right)^{m_B} \cdot \left(\frac{\theta}{\theta_0} \right)^n, \tag{8.66}$$

where $\dot{\gamma}_0 \equiv 1/\text{s}$ is a constant introduced for dimensional consistency, $p_B = -[(\mathbf{T}_B)_{11} + (\mathbf{T}_B)_{22} + (\mathbf{T}_B)_{33}]/3$ is the hydrostatic pressure, $R(x) = (x + |x|)/2$ is the ramp function, and $\hat{\tau}_B$, β, m_B, n, and θ_0 are specified material parameters. In this framework, the temperature dependence of the flow rate is taken to follow a power law form. In summary, the velocity gradient of the viscoelastic flow of network B can be written

$$\dot{\mathbf{F}}_B^v = \dot{\gamma}_B \mathbf{F}_B^{e-1} \frac{\text{dev}[\mathbf{T}_B]}{\tau_B} \mathbf{F}. \tag{8.67}$$

The TNM model requires the material parameters in Table 8.3.

Table 8.3 Material Parameters Used by the Three Network Model

Index	Symbol	Description
1	μ_A	Shear modulus of network A
2	$\hat{\theta}$	Temperature factor
3	λ_L	Locking stretch
4	κ	Bulk modulus
5	$\hat{\tau}_A$	Flow resistance of network A
6	a	Pressure dependence of flow
7	m_A	Stress exponential of network A
8	n	Temperature exponential
9	μ_{Bi}	Initial shear modulus of network B
10	μ_{Bf}	Final shear modulus of network B
11	β	Evolution rate of μ_B
12	$\hat{\tau}_B$	Flow resistance of network B
13	m_B	Stress exponential of network B
14	μ_C	Shear modulus of network C
15	α	Thermal expansion coefficient
16	θ_0	Thermal expansion reference temperature

8.6.1 Matlab Implementation of the Three Network Model

Like the other models in this chapter, the TNM is formulated as a set of differential equations. These equations can be quickly solved for uniaxial loading using the equations in the following function (Figures 8.37 and 8.38).

A Matlab function that exercises the `mat_TNM()` function is shown in the following code, and the results from running the code are shown in Figure 8.39.

8.6.2 Python Implementation of the Three Network Model

For incompressible uniaxial loading the Three Network (TN) model can be implemented into Python code as shown below. The code builds upon the code from Chapter 5, and previous examples from this chapter. First a few help functions are defined.

Additional Code to "Polymer_Mechanics_Chap09.py":

```
def Dev(A):
    """Deviatoric part of a tensor"""
    return A - sum(A)/3.0

def Inv(A):
    """Inverse of a tensor"""
    return array([1.0, 1.0, 1.0]) / A

def pressure(A):
    """Pressure of a stress tensor"""
    return -sum(A) / 3.0
```

The time-derivative of the state variables in the TN model are calculated in the function `TNM_timeDer_3D()`, listed below. The main function for the TN model is provided by the function `TNM_3D()`.

The actual commands that sets everything up and calls the main function are listed in the file `TNM_Compressible_Uniaxial.py`.

The results from running this code are shown in Figure 8.40.

Additional Code to "Polymer_Mechanics_Chap09.py":

```
def TNM_timeDer_3D(statev, t, params, time0, time1, F0, F1):
    """Returns statevDot"""
    muA, lamL, kappa = params[0:3]
    tauHatA, a, mA = params[3:6]
    muBi, muBf, beta = params[6:9]
    tauHatB, mB, muC = params[9:12]
    res = zeros(len(statev))
    F = F0 + (t-time0) / (time1-time0) * (F1-F0)

    # Network A: FAv
    Fv = statev[0:3]
    muB = statev[6]
    Fe = F / Fv
    Stress = toVec(EC_3D(Fe, [muA, lamL, kappa]))
    tau = norm(Dev(Stress)) + 1.0e-9
    gamDot = (tau / (tauHatA + a * ramp(pressure(Stress))))**mA
    res[0:3] = gamDot/tau * (Inv(Fe) * Dev(Stress) * F)
    res[6] = -beta * (statev[6] - muBf) * gamDot

    # Network B: FBv
    Fv = statev[3:6]
    muB = statev[6]
    Fe = F / Fv
    Stress = toVec(EC_3D(Fe, [muB, lamL, kappa]))
    tau = norm(Dev(Stress)) + 1.0e-9
    gamDot = (tau / (tauHatB + a * ramp(pressure(Stress))))**mB
    res[3:6] = gamDot/tau * (Inv(Fe) * Dev(Stress) * F)
    return res
```

Additional Code to "Polymer_Mechanics_Chap09.py":

```
def TNM_3D(F0, F1, statev0, time0, time1, params):
    """TN-model. 3D loading specified by principal stretches.
       params: [muA, lamL, kappa, tauHatA, a, mA, muBi, mBf,
                beta, tauHatB, mB, muC].
       Returns: (true stress, statev1)"""
    muA, lamL, kappa, s = params[0:4]
    muC = params[11]
    StressC = toVec(EC_3D(F1, [muC, lamL, kappa]))

    statev1 = scipy.integrate.odeint(TNM_timeDer_3D, statev0, \
            [time0, time1], args=(params, time0, time1, F0, F1))[1]
    FAv1 = statev1[0:3]
    FBv1 = statev1[3:6]
    muB = statev1[6]

    FAe1 = F1 / FAv1
    StressA = toVec(EC_3D(FAe1, [muA, lamL, kappa]))

    FBe1 = F1 / FBv1
    StressB = toVec(EC_3D(FBe1, [muB, lamL, kappa]))

    Stress = StressA + StressB + StressC
    return (Stress, statev1)
```

Python Code "TNM_compressible_uniaxial.py":

```
from Polymer_Mechanics_Chap05 import *
from Polymer_Mechanics_Chap09 import *

N = 50
timeVec = append(linspace(0,10.0,N), linspace(10.0,20.0,N)[1:])
trueStrain = append(linspace(0,0.2,N), linspace(0.2,0,N)[1:])

params = [290.0, 5.0, 2000.0, 7.0, 0.0, 9.5, \
        130.0, 50.0, 10.0, 24.0, 9.0, 8.0]

# FAv, FBv, muB
statev0 = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, params[6]]

trueStress = uniaxial_stress_visco(TNM_3D, timeVec, trueStrain,
params, statev0)

plot(trueStrain, trueStress, 'r-', label='Python Calculation')
xlabel('True Strain')
ylabel('True Stress (MPa)')
grid('on')
savefig('TNM_compressible_uniaxial.png', dpi=300)
show()
```

Matlab File Name: mat_TNM.m

```
function [stress] = mat_TNM(time, strain, params)
%mat_TNM Thre-Network Model
%Incompressible uniaxial loading (uses true strain & stress)
%Parameters: [muA lambdaL tauHatA a mA muBi muBf beta tauHatB mB muC]
%State variabes: [lambdaAv lambdaBv muB]
muA     = params( 1);
lambdaL = params( 2);
muBi    = params( 6);
muC     = params(11);
stress = 0 * strain;
state = [1 1 muBi]';
for i = 2 : length(strain)
    [t,stateAll] = ode45(@(t,y) flow(t, y, time, strain, params), time(i-1:i), state);
    state = stateAll(end,:);
    lambda = exp(strain(i));
    stressA = mat_EC(time(i), log(lambda/state(1)), [muA lambdaL]);
    stressB = mat_EC(time(i), log(lambda/state(2)), [state(3) lambdaL]);
    stressC = mat_EC(time(i), strain(i), [muC lambdaL]);
    stress(i) = stressA + stressB + stressC;
end
end

function res = flow(time, state, timeVec, strainVec, params)
% Calculates the time-derivative of the state variables
muA     = params( 1);
lambdaL = params( 2);
tauHatA = params( 3);
a       = params( 4);
mA      = params( 5);
muBf    = params( 7);
beta    = params( 8);
tauHatB = params( 9);
mB      = params(10);
% network A
lambda = exp( interp1(timeVec, strainVec, time) ); % stretch at the specified time
lambdaAv = state(1);
stressA = mat_EC(time, log(lambda/lambdaAv), [muA lambdaL]);
gammaADot = (abs(stressA)/(tauHatA+a*stressA))^mA;
res(1,1) = gammaADot * sign(stressA) * lambdaAv;
% network B
lambdaBv = state(2);
muB = state(3);
stressB = mat_EC(time, log(lambda/lambdaBv), [muB lambdaL]);
res(2,1) = (abs(stressB)/(tauHatB+a*stressB))^mB * sign(stressB) * lambdaBv;
res(3,1) = -beta * (muB - muBf) * gammaADot;
end
```

Figure 8.37 Matlab implementation of the Three Network Model.

Matlab File Name: test_mat_TNM.m

```
time = linspace(0,2)';
strain = [linspace(0, 0.5,50)'; linspace(0.5, 0, 50)'];
stress = mat_TNM(time, strain, [192 3.11 7.2 0.0001 9.7 131 48 11 25 10 8]);
hp = plot(strain,stress);
```

Figure 8.38 Matlab code to test the implementation of the Three Network Model.

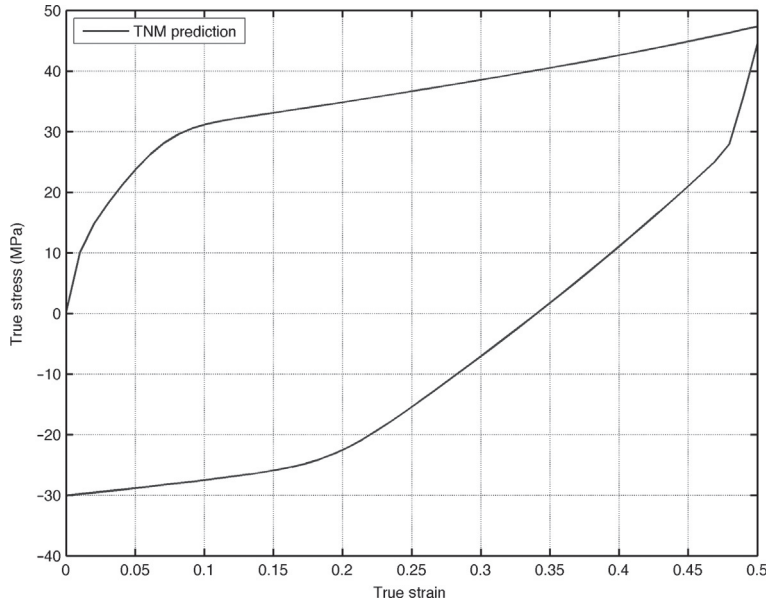


Figure 8.39 Matlab test of the Three Network Model.

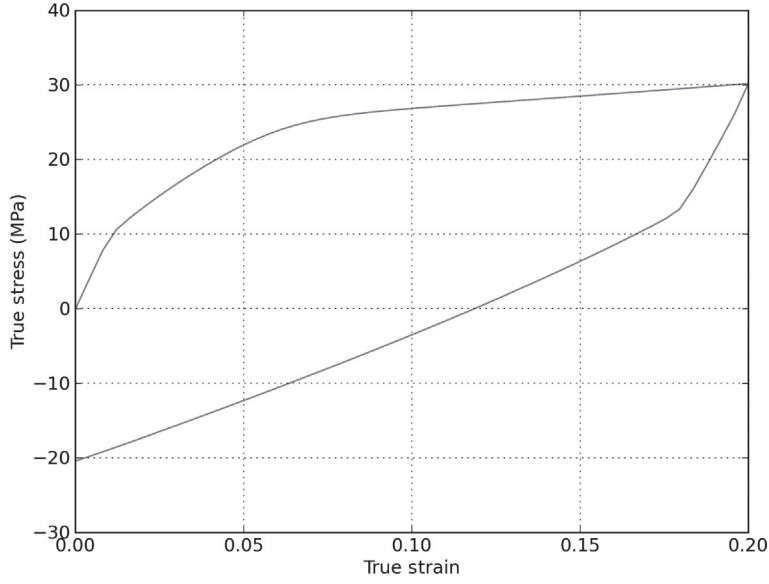


Figure 8.40 Predictions from the Python file `TNM_Compressible_Uniaxial.py`.

8.6.3 Use of the Three Network Model for Polymer Modeling

The TNM has many similarities to the HM in terms of target materials and behaviors. The TNM is specifically useful for predicting the mechanical response of thermoplastic materials below the glass transition temperature, or semi-crystalline polymers below the melting temperature.

Figures 8.41–8.43 compare the predictions from the TNM to experimental data for UHMWPE (crosslinked GUR 1050). As shown in the figures the TNM accurately predicts the behavior of the material.

Figure 8.41 shows the response in uniaxial tension at three different strain-rates. Figure 8.42 shows the response in cyclic loading at an intermediate strain, and Figure 8.43 shows the predictions in unaxial compression.

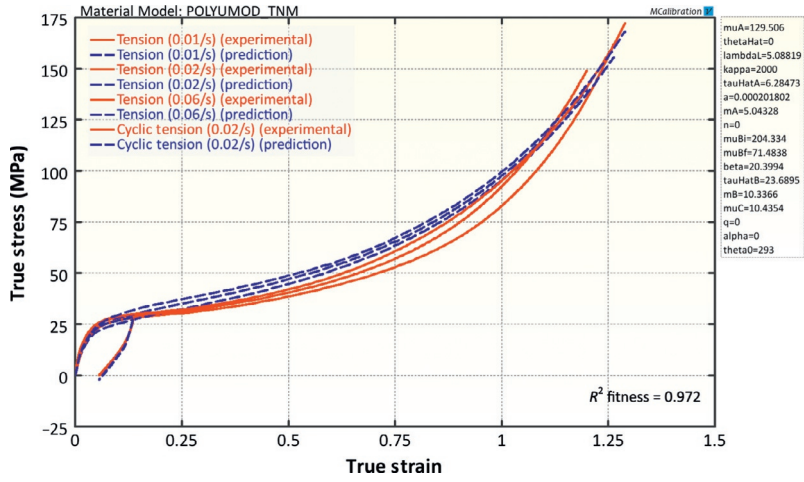


Figure 8.41 Uniaxial tensile results from the Three Network Model.

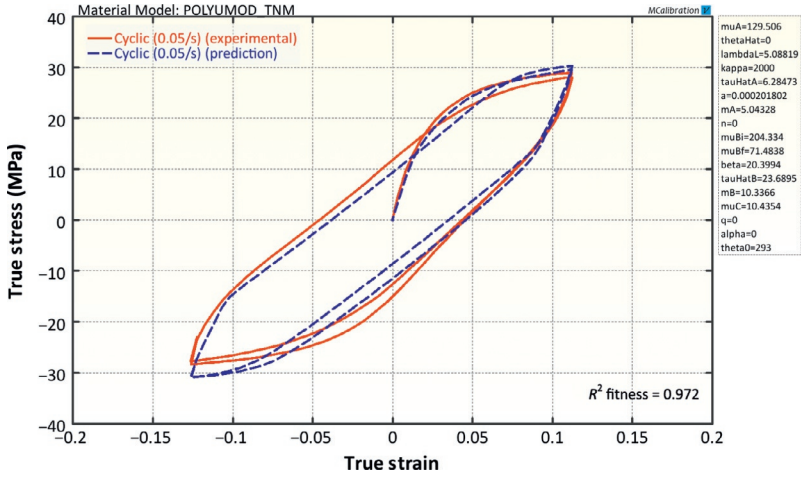


Figure 8.42 Uniaxial cyclic results from the Three Network Model.

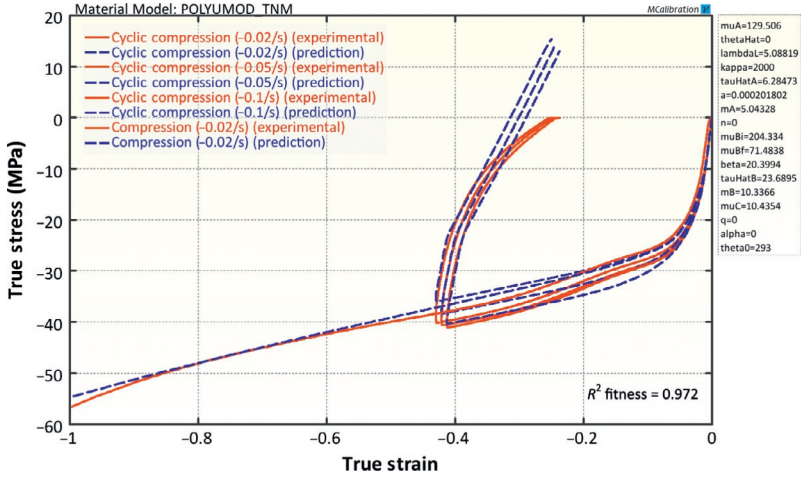


Figure 8.43 Uniaxial compressive results from the Three Network Model.

8.7 Parallel Network Model

The Parallel Network (PN) model is an extension of the TNM that allows for an arbitrary number of parallel networks, where each network consists of an elastic component and an optional flow component. The PN model is supported by both the

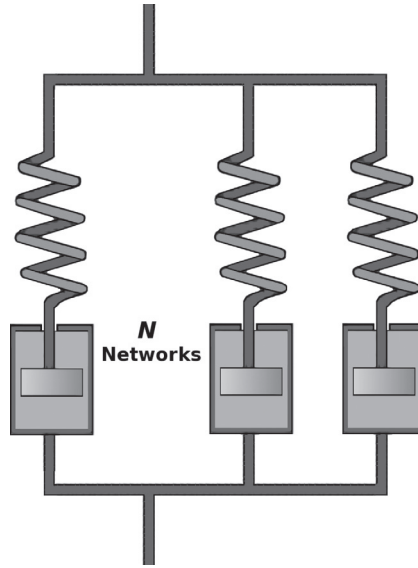


Figure 8.44 Rheological representation of the parallel network model.

MCalibration [30] and the PolyUMod [22] software. Figure 8.44 shows a rheological representation of the PN model.

The constitutive model framework supports a large number of different isotropic and anisotropic elastic behaviors, coupled with a large collection of isotropic and anisotropic viscoplastic behaviors. The model framework also supports many different damage and failure models. These features make this model framework one of the most advanced material model frameworks that have been developed, and it can capture the mechanical response of almost any polymeric material. This flexibility can make the model challenging to setup for someone not experienced in visualizing the response of rheological models. For that reason this section contains a selection of simple model frameworks and their corresponding responses. A complete description is provided in the PolyUMod User's Manual.

Figure 8.45 shows the stress-strain response for a simple one-network model consisting of a Neo-Hookean element. As shown in this figure the material response is independent of the applied strain rate.

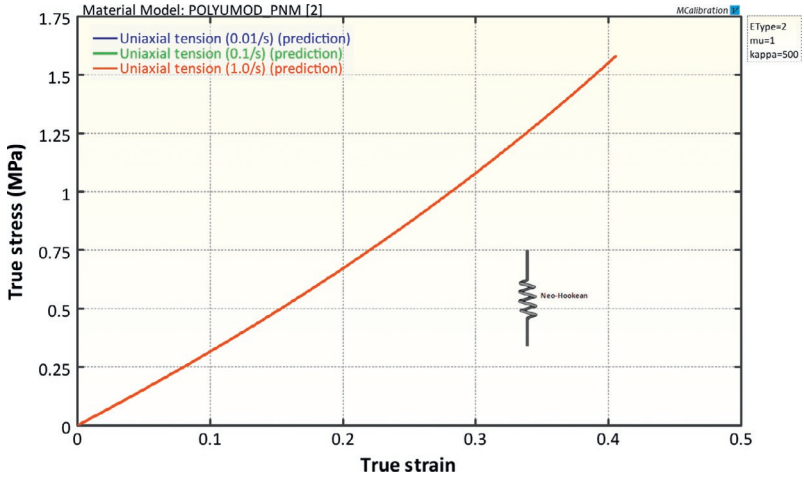


Figure 8.45 Predictions from a simple PN model with hyperelastic network.

By adding a viscoelastic flow element with a flow rate that depends on the normalized shear stress raised to a power:

$$\dot{\gamma} = \left(\frac{\tau}{\hat{\tau}}\right)^m, \tag{8.68}$$

the material response becomes non-linear viscoelastic once the stress becomes sufficiently large. The predicted stress-strain response of this model framework is shown in Figure 8.46.

The stress-strain response after yielding can be modified by adding a second network (see Figure 8.47. This second network is typically less stiff than the first network, and makes the model similar to the BB model discussed in Section 8.2.

Some polymers soften after yielding. The experimentally observed softening can often be captured using a yield evolution equation that control how the flow resistance evolves with plastic strain. Figure 8.48 illustrates the predicted stress-strain response when the flow resistance ($\hat{\tau}$) depends exponentially on the plastic strain:

$$\hat{\tau} = \hat{\tau}_0 \left\{ f_f + (1 - f_f) \exp\left[\frac{-\varepsilon_p}{\hat{\varepsilon}}\right] \right\}, \tag{8.69}$$

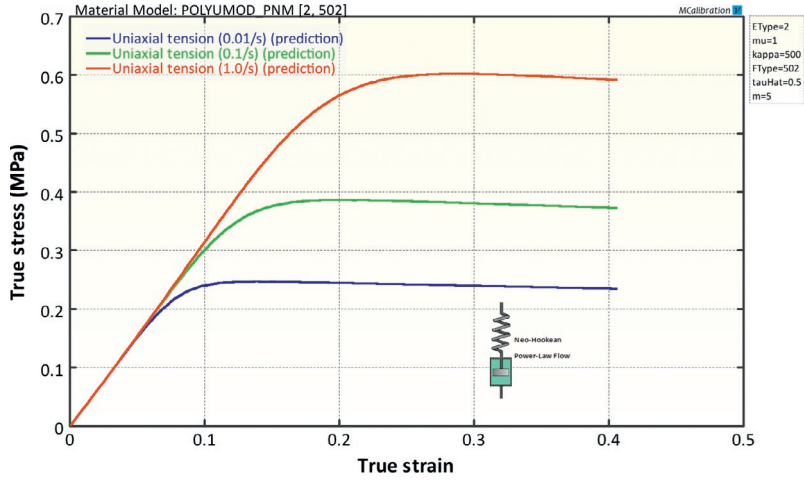


Figure 8.46 Predictions from a simple PN model with a Neo-Hookean hyperelastic component in series with a power-law flow element.

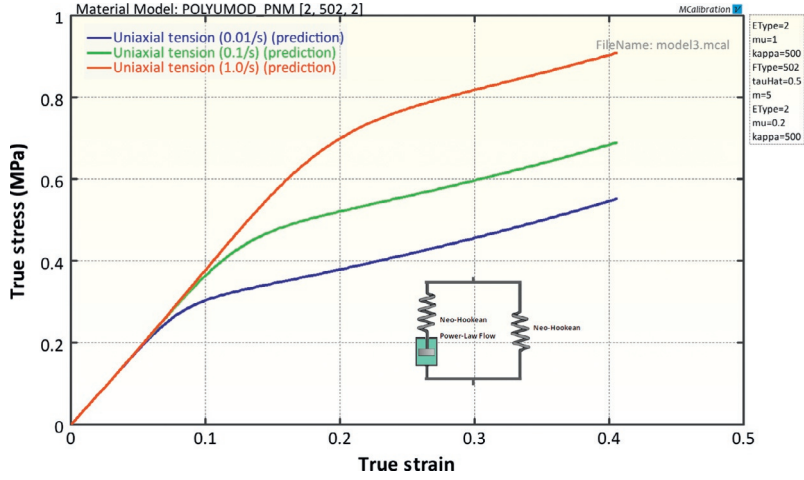


Figure 8.47 Predictions from a simple PN model with two parallel networks. The second network gives the stiffening beyond the onset of viscoplastic flow.

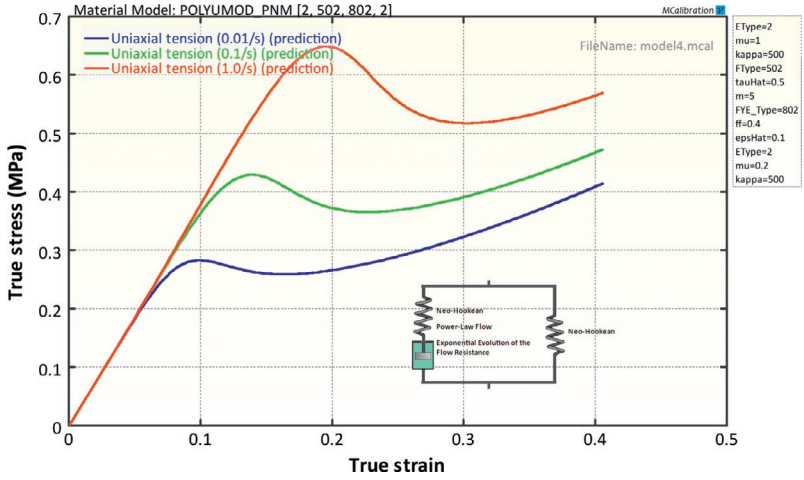


Figure 8.48 Predictions from a simple PN model with two parallel networks. The flow element has an exponential evolution of the flow resistance.

where f_f is the final value that $\hat{\tau}$ evolves to, $\hat{\varepsilon}$ is the characteristic strain for the evolution, and ε_p is the effective plastic strain.

8.8 Use of Viscoplasticity in Polymer Modeling

Viscoplasticity is the most accurate material model framework available to represent the mechanical response of all polymers. A large collection of viscoplastic material models have been developed in the academic literature, and this chapter reviewed a collection of commonly used models that have been shown to accurately capture the behavior of elastomers, thermoplastics, and other polymers.

The material models presented here have been designed to be easy to use and calibrate. In almost all cases the hyperelastic portion of the response is limited to I_1-based energy functions, making it possible to calibrate the material models to only uniaxial tension and/or compression data. To capture the viscous time-dependent response also requires that the material be tested at different strain rates and preferably with loading—

hold—unloading cycles. Multiple examples of the practical use of viscoplastic material models are given in Chapter 11.

8.9 Python Code Examples

The behavior of a number of the viscoplastic material models presented in this chapter were examined using short Python functions. In each example, it was listed "Additional Code to Polymer_Mechanics_Chap09.py". This section summarizes the file `Polymer_Mechanics_Chap09.py`. This file can also be downloaded from this web address:
http://PolymerMechanics.com/Polymer_Mechanics_Chap09.zip.

```
from Polymer_Mechanics_Chap05 import *
import scipy.integrate

def ramp(x):
    return (x + abs(x)) / 2.0

def toVec(A):
    """Convert a 3x3 matrix to vector"""
    return array([A[0][0], A[1][1], A[2][2]])

def Dev(A):
    """Deviatoric part of a tensor"""
    return A - sum(A)/3.0

def Inv(A):
    """Inverse of a tensor"""
    return array([1.0, 1.0, 1.0]) / A

def pressure(A):
    """Pressure of a stress tensor"""
    return -sum(A) / 3.0

def uniaxial_stress_visco(model, timeVec, trueStrainVec, params):
    """Compressible uniaxial loading. Returns true stress."""
    stress = zeros(len(trueStrainVec))
    lam2_1 = 1.0
    FBv1 = array([1.0, 1.0, 1.0])
    for i in range(1, len(trueStrainVec)):
        print 'uniaxial_stress: i=', i, ' of ', len(trueStrainVec)
        time0 = timeVec[i-1]
        time1 = timeVec[i]
        lam1_0 = exp(trueStrainVec[i-1])
        lam1_1 = exp(trueStrainVec[i])
        lam2_0 = lam2_1
        F0 = array([lam1_0, lam2_0, lam2_0])
        F1 = array([lam1_1, lam2_1, lam2_1])
        FBv0 = FBv1.copy()
        calcS22Abs = lambda x: abs(model(F0, array([lam1_1,x,x]), \
                FBv0, time0, time1, params)[0][1])
        # search for transverse stretch that gives S22=0
        lam2_1 = scipy.optimize.fmin(calcS22Abs, x0=lam2_0,
                                     xtol=1e-9, ftol=1e-9, disp=False)
        res = model(F0, array([lam1_1, lam2_1, lam2_1]), FBv0, time0,
```

```
        time1, params)
        stress[i] = res[0][0]
        FBv1 = res[1]
    return stress

def BB_timeDer_3D(Fv, t, params, time0, time1, F0, F1):
    """Returns FvDot"""
    mu, lamL, kappa, s, xi, C, tauBase, m, tauCut = params[:9]
    F = F0 + (t-time0) / (time1-time0) * (F1-F0)
    Fe = F / Fv
    Stress = toVec(EC_3D(Fe, [s*mu, lamL, kappa]))
    devStress = Stress - sum(Stress)/3
    tau = norm(devStress)
    lamCh = sqrt(sum(Fv*Fv)/3.0)
    lamFac = lamCh - 1.0 + xi
    gamDot = lamFac**C * (ramp(tau/tauBase-tauCut)**m)
    prefac = 0.0
    if tau > 0: prefac = gamDot / tau
    FeInv = array([1.0, 1.0, 1.0]) / Fe
    FvDot = prefac * (FeInv * devStress * F)
    return FvDot

def BB_3D(F0, F1, FBv0, time0, time1, params):
    """BB-model. 3D loading specified by principal stretches.
       params: [muA, lamL, kappa, s, xi, C, tauHat, m, tauCut].
       Returns: (true stress, FBv1)"""
    muA, lamL, kappa, s = params[:4]
    StressA = toVec(EC_3D(F1, [muA, lamL, kappa]))
    FBv1 = scipy.integrate.odeint(BB_timeDer_3D, FBv0, \
            [time0, time1], args=(params, time0, time1, F0, F1))[1]
    FBe1 = F1 / FBv1
    StressB = toVec(EC_3D(FBe1, [s*muA, lamL, kappa]))
    Stress = StressA + StressB
    return (Stress, FBv1)

def TNM_timeDer_3D(statev, t, params, time0, time1, F0, F1):
    """Returns statevDot"""
    muA, lamL, kappa = params[0:3]
    tauHatA, a, mA = params[3:6]
    muBi, muBf, beta = params[6:9]
    tauHatB, mB, muC = params[9:12]
    res = zeros(len(statev))
    F = F0 + (t-time0) / (time1-time0) * (F1-F0)

    # Network A: FAv
    Fv = statev[0:3]
    muB = statev[6]
    Fe = F / Fv
    Stress = toVec(EC_3D(Fe, [muA, lamL, kappa]))
    tau = norm(Dev(Stress)) + 1.0e-9
    gamDot = (tau / (tauHatA + a * ramp(pressure(Stress))))**mA
    res[0:3] = gamDot/tau * (Inv(Fe) * Dev(Stress) * F)
    res[6] = -beta * (statev[6] - muBf) * gamDot

    # Network B: FBv
    Fv = statev[3:6]
    muB = statev[6]
    Fe = F / Fv
    Stress = toVec(EC_3D(Fe, [muB, lamL, kappa]))
    tau = norm(Dev(Stress)) + 1.0e-9
    gamDot = (tau / (tauHatB + a * ramp(pressure(Stress))))**mB
    res[3:6] = gamDot/tau * (Inv(Fe) * Dev(Stress) * F)
    return res

def TNM_3D(F0, F1, statev0, time0, time1, params):
    """TN-model. 3D loading specified by principal stretches.
```

```
    params: [muA, lamL, kappa, tauHatA, a, mA, muBi, mBf,
             beta, tauHatB, mB, muC].
    Returns: (true stress, statev1)"""
muA, lamL, kappa, s = params[0:4]
muC = params[11]
StressC = toVec(EC_3D(F1, [muC, lamL, kappa]))

statev1 = scipy.integrate.odeint(TNM_timeDer_3D, statev0, \
        [time0, time1], args=(params, time0, time1, F0, F1))[1]
FAv1 = statev1[0:3]
FBv1 = statev1[3:6]
muB = statev1[6]

FAe1 = F1 / FAv1
StressA = toVec(EC_3D(FAe1, [muA, lamL, kappa]))

FBe1 = F1 / FBv1
StressB = toVec(EC_3D(FBe1, [muB, lamL, kappa]))

Stress = StressA + StressB + StressC
return (Stress, statev1)
```

8.10 Exercises

1. Explain the physical reason why elastomers are viscoelastic.
2. Why cannot a linear viscoelastic material model predict the strain amplitude dependence of the storage and loss moduli, but the BB model can?
3. Calibration of the BB model
   - Download the experimental data file `neoprene.csv` from the PolymerFEM.com website: http://polymerfem.com/polymer_files/Nitrile_rubber.zip
   - Plot the experimental data
   - Find the material parameters for the BB-model
   - Plot the experimental data and the model predictions in one figure
   - How well does the model work?
   - In what range is the model accurate?
4. Write a Matlab function file for the BB model with Ogden-Roxburgh Mullins effect.
5. What are the main features of the AB viscoplastic material model?

6. The DNF model has been specifically designed for fluoropolymers. What aspects of the model make it more suitable than, say, the AB viscoplastic model?
7. The TNM is an exceptionally versatile material model that can predict the non-linear viscoplastic response of most thermoplastics. What materials are the TN model not suitable for?

References

[1] J.S. Bergström, M.C. Boyce. Constitutive modeling of the large strain time-dependent behavior of elastomers. J. Mech. Phys. Solids, 46 1998 931-954.
[2] J.S. Bergström, M.C. Boyce. Mechanical behavior of particle filled elastomers. Rubber Chem. Technol., 72 1999 633-656.
[3] J.S. Bergström. Large Strain Time-Dependent Behavior of Elastomeric Materials. Ph.D. thesis, MIT 1999.
[4] J.S. Bergström, M.C. Boyce. Large strain time-dependent behavior of filled elastomers. Mech. Mater., 32 2000 620-644.
[5] J.S. Bergström, M.C. Boyce. Constitutive modeling of the time-dependent and cyclic loading of elastomers and application to soft biological tissues. Mech. Mater., 33 2001 523-530.
[6] M.S. Green, A.V. Tobolsky. A new approach to the theory of relaxing polymeric media. J. Chem. Phys., 14 (2) 1946 80-92.
[7] A.R. Johnson, C.J. Quigley. A viscohyperelastic Maxwell model for rubber viscoelasticity. Rubber Chem. Technol., 65 1992 137-153.
[8] A.R. Johnson, R.G. Stacer. Rubber viscoelasticity using the physically constrained system's stretches as internal variables. Rubber Chem. Technol., 66 1993 567-577.
[9] A.R. Johnson, C.J. Quigley, C.E. Freese. A viscohyperelastic finite element model for rubber. Comput. Methods Appl. Mech. Eng., 127 1995 163-180.
[10] C.M. Roland. Network recovery from uniaxial extension I: elastic equilibrium. Rubber Chem. Technol., 62 1989 863-879.
[11] C.M. Roland, M.L. Warzel. Orientation effects in rubber double networks. Rubber Chem. Technol., 63 1990 285-297.

[12] ANSYS, Inc.. ANSYS Mechanical, Release 15.0, 2014.
[13] MSC Software. Marc 2014, 2014.
[14] LSTC Inc.. LS-DYNA V971, 2014.
[15] HKS, Inc.. ABAQUS, Pawtucket, RI, ver. 6.14, 2014.
[16] M.C. Boyce, G.G. Weber, D.M. Parks. On the kinematics of finite strain plasticity. J. Mech. Phys. Solids, 37 (5) 1989 647-665.
[17] P.G. de Gennes. Reptation of a polymer chain in the presence of fixed obstacles. J. Chem. Phys., 55 (2) 1971 572-579.
[18] M. Doi, S.F. Edwards. The Theory of Polymer Dynamics. Oxford University Press, Oxford, 1986.
[19] E.M. Arruda, M.C. Boyce. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. J. Mech. Phys. Solids, 41 (2) 1993 389-412.
[20] E.M. Arruda, M.C. Boyce. Evolution of plastic anisotropy in amorphous polymers during finite straining. Int. J. Plasticity, 9 1993 697-720.
[21] E.M. Arruda, M.C. Boyce. Effects of strain rate, temperature and thermomechanical coupling on the finite strain deformation of glassy polymers. Mech. Mater., 19 1995 193-212.
[22] PolyUMod. http://PolyUMod.com/.
[23] O.A. Hasan, M.C. Boyce. Energy storage during inelastic deformation of glassy polymers. Polymer, 34 (24) 1993 5085-5092.
[24] J.S. Bergström, C.M. Rimnac, S.M. Kurtz. Prediction of multiaxial mechanical behavior for conventional and highly crosslinked UHMWPE using a hybrid constitutive model. Biomaterials, 24 2003 1365-1380.
[25] K. Ho, E. Krempl. Extension of the viscoplastic theory based on overstress (VBO) to capture non-standard rate dependence in solids. Int. J. Plasticity, 18 2002 851-872.
[26] A. Lion. On the large deformation behavior of reinforced rubber at different temperatures. J. Mech. Phys. Solids, 45 1997 1805-1834.
[27] A. Khan, H. Zhang. Finite deformation of a polymer: experiments and modeling. Int. J. Plasticity, 17 (9) 2001 1167-1188.
[28] T. Kletschkowski, U. Schomburg, A. Bertram. Endochronic viscoplastic material models for filled PTFE. Mech. Mater., 34 (12) 2002 795-808.
[29] J.S. Bergström, S.M. Kurtz, C.M. Rimnac, A.A. Edidin. Constitutive modeling of ultra-high molecular weight polyethylene under large-deformation and cyclic loading conditions. Biomaterials, 23 2002 2329-2343.
[30] Veryst Engineering. http://PolyUMod.com/.