AE 681 COMPOSITE MATERIALS PROJECT REPORT

by

Rohan Sai Adiserla - 241010051
P L N S Sumanth - 241010043

DEPARTMENT OF AEROSPACE ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April 2025

Objective: To find the effective properties of the composite using:

1. Strength of Materials Approach
2. Hill's Concentration Factors Approach
   - Voigt Approximation
   - Reuss Approximation
3. Concentric Cylinder Assemblage Model
4. Self-consistent Method
5. Mori-Tanaka Method
6. Halpin-Tsai method
7. Hashin Shtrikman Bounds

Given Materials:

Fiber type	**E-glass 21xK43 Gevetex**
Longitudinal modulus, E_1 (GPa)	*80*
Transverse modulus, E_2 (GPa)	*80*
In-plane shear modulus, G_{12} (GPa)	*33.33*
Major Poisson's ratio, ν_{12}	*0.2*
Transverse shear modulus, G_{23} (GPa)	*33.33*
Longitudinal thermal coefficient, α_1 (10−6 /°C)	*4.9*
Transverse thermal coefficient, α_2 (10−6 /°C)	*4.9*

Matrix type	**3501-6 epoxy**
Modulus, E_m (GPa)	*4.2*
Poisson's ratio, ν_m	*0.34*
Thermal coefficient, α_m (10−6 /°C)	*45*

Strength of Materials Approach

In the Strength of Materials approach, we utilize direct formulas derived from compatibility conditions and equilibrium requirements. These equations provide a straightforward method to determine the effective properties of composite materials.

The final equations for calculating the effective properties of the composite are presented below,

Effective Axial Modulus, E_1^* :

$$E_1^* = E_1^{(f)} V_f + E^{(m)} \left(1 - V_f\right)$$

Effective Axial (Major) Poison's Ratio, v_{12}^* :

$$v_{12}^{(*)} = v_{12}^{(f)} V_f + v^{(m)} \left(1 - V_f\right)$$

Effective Transverse Modulus, E_2^* :

$$\frac{1}{E_2^*} = \frac{V_f}{E_2^{(f)}} + \frac{\left(1 - V_f\right)}{E^{(m)}}$$

Effective Axial Shear Modulus, G_{12}^* :

$$\frac{1}{G_{12}^*} = \frac{V_f}{G_{12}^{(f)}} + \frac{\left(1 - V_f\right)}{G^{(m)}}$$

Effective Coefficient of Thermal Expansion α_1^* :

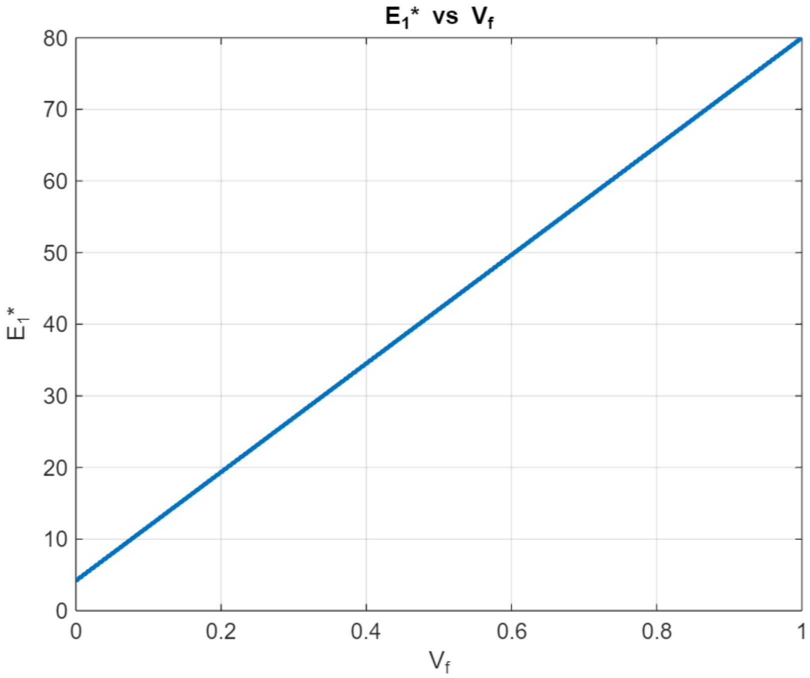
$$\alpha_1^* = \frac{\left(\alpha_1^{(f)} E_1^{(f)} - \alpha^{(m)} E^{(m)}\right) V_f + \alpha^{(m)} E^{(m)}}{\left(E_1^{(f)} - E^{(m)}\right) V_f + E^{(m)}}$$

Effective Coefficient of Thermal Expansion α_2^* :

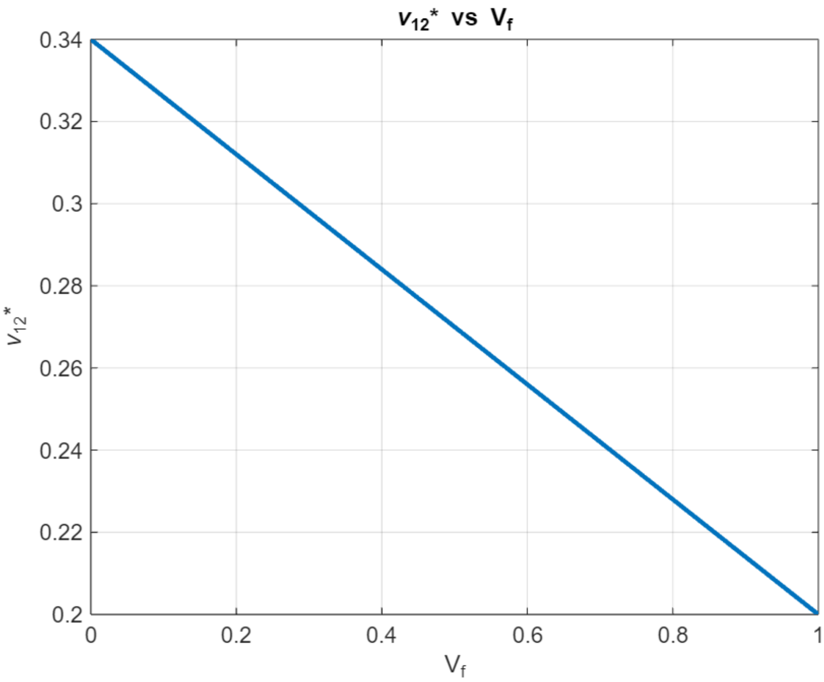
$$\alpha_2^* = \alpha_2^{(f)} V_f + \alpha^{(m)} V_m + \left(\frac{E_1^{(f)} v^{(m)} - E^{(m)} v_{12}^{(f)}}{E_1^*} \right) \left(\alpha^{(m)} - \alpha_1^{(f)} \right) (1 - V_f) V_f$$

Results:

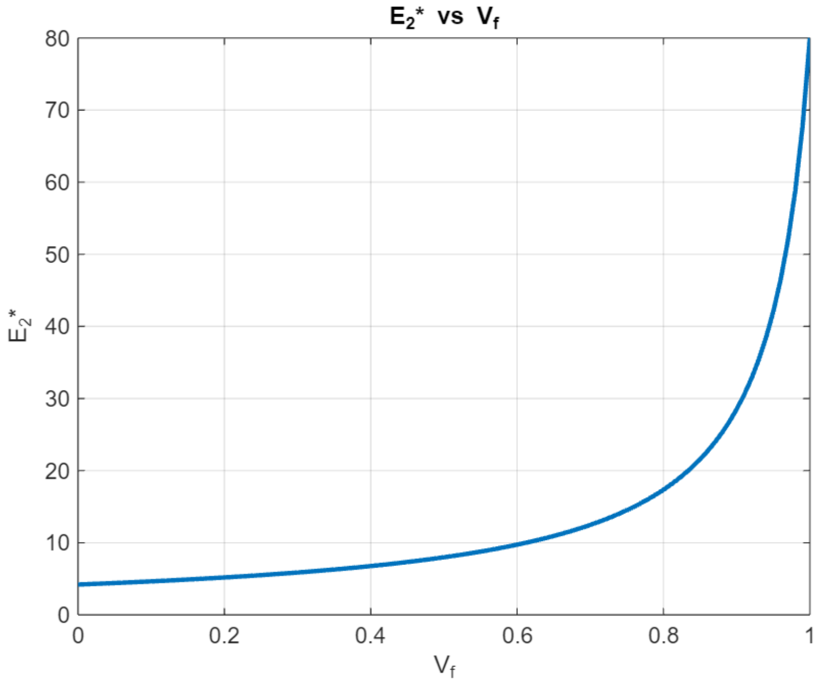
Effective Axial Modulus, E_1^* :



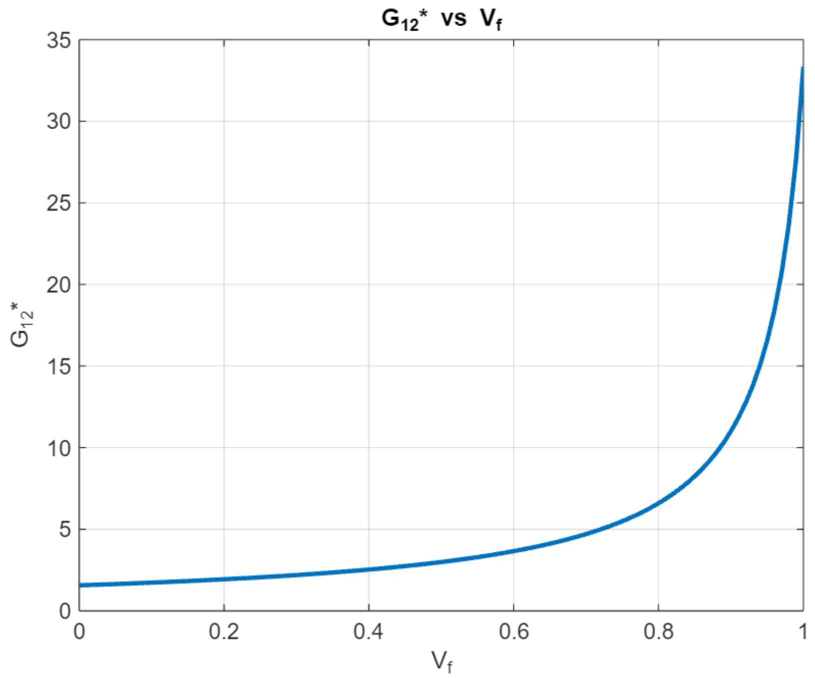
Effective Axial (Major) Poison's Ratio, v_{12}^* :



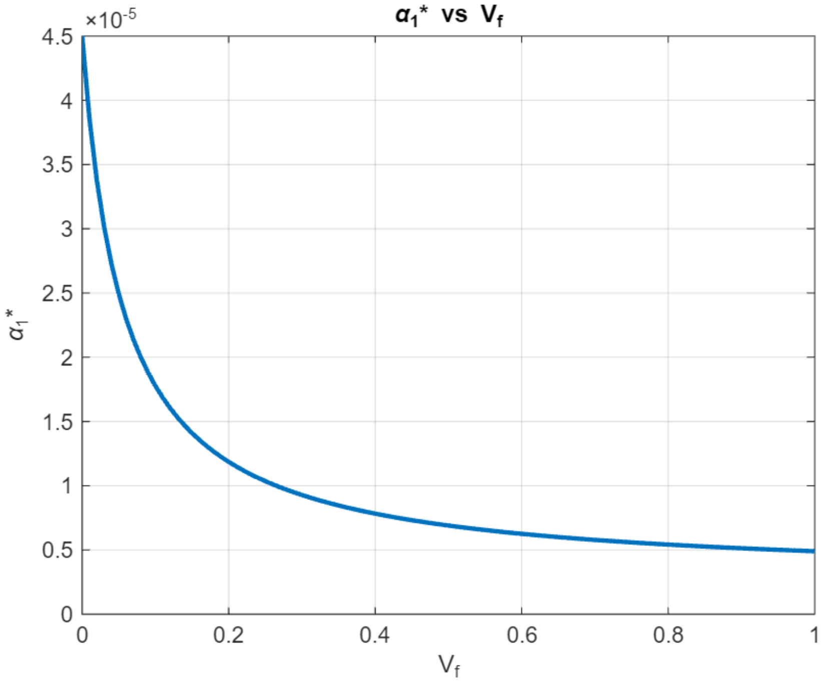
Effective Transverse Modulus, E_2^* :



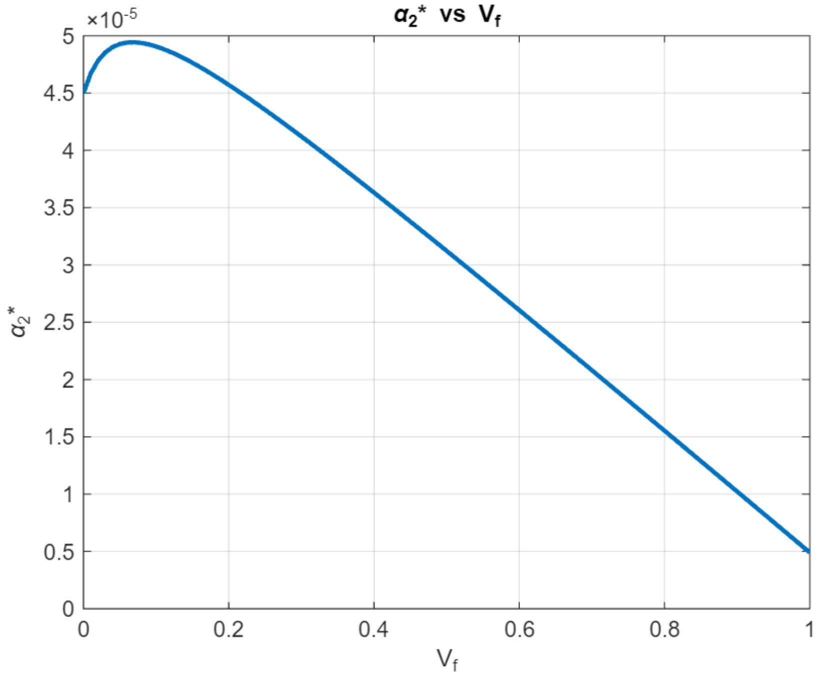
Effective Axial Shear Modulus, G_{12}^* :



Effective Coefficients of Thermal Expansion α_1^* :



Effective Coefficients of Thermal Expansion α_2^* :



Hill’s Concentration Factors Approach

We relate the phase-averaged strains and stresses to composite average values through concentration factors and then substitute these relationships into the expressions for composite stress and strain. We then identify the constraints imposed by volume averaging, which show that the volume fraction weighted concentration factors must sum to the identity tensor. By applying these constraints to our composite stress-strain relationships, we end up with equations that express the effective stiffness and compliance tensors in terms of constituent phase properties and their concentration factors.

$$C^{*}_{ijkl} = C^{(m)}_{ijkl} + V_f \left(C^{(f)}_{ijpq} - C^{(m)}_{ijpq} \right) A^{(f)}_{pqkl}$$

$$S_{ijkl} = S^{(m)}_{ijkl} + V_f \left(S^{(f)}_{ijrs} - S^{(m)}_{ijrs} \right) B^{(f)}_{rskl}$$

Voigt Approximation:

Voigt assumed that the strains are constants throughout the composite. Thus, we can say that,

$$A^{(f)}_{ijkl} = A^{(m)}_{ijkl} = I_{ijkl}$$

Doing that we end up with,

$$C^{*}_{ijkl} = C^{(m)}_{ijkl} + V_f \left(C^{(f)}_{ijpq} - C^{(m)}_{ijpq} \right)$$

Reuss Approximation:

Reuss assumed that the stresses are constant throughout the composite. This assumption leads to the relation,

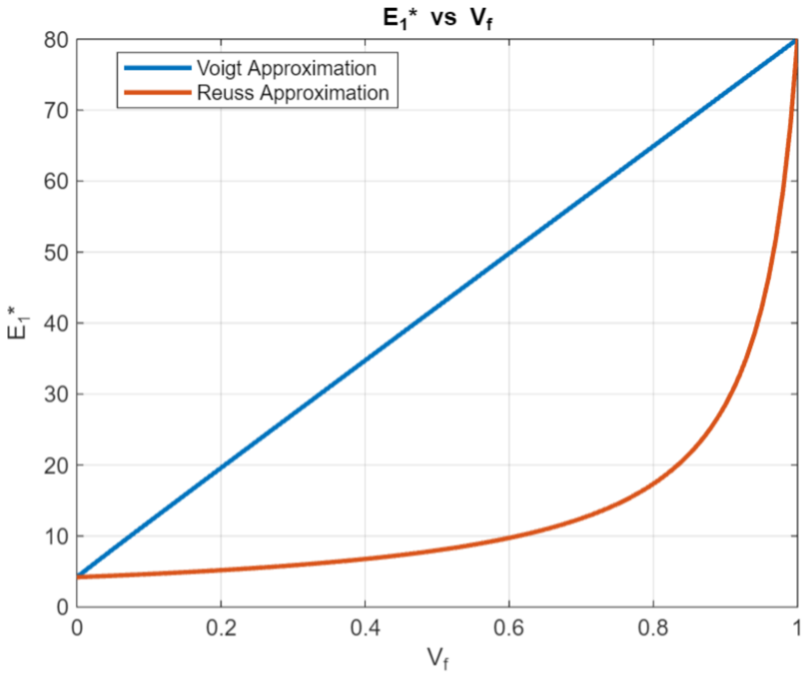
$$\mathrm{B}^{(f)}_{ijkl} = \mathrm{B}^{(m)}_{ijkl} = I_{ijkl}$$

Substituting that we end up with,

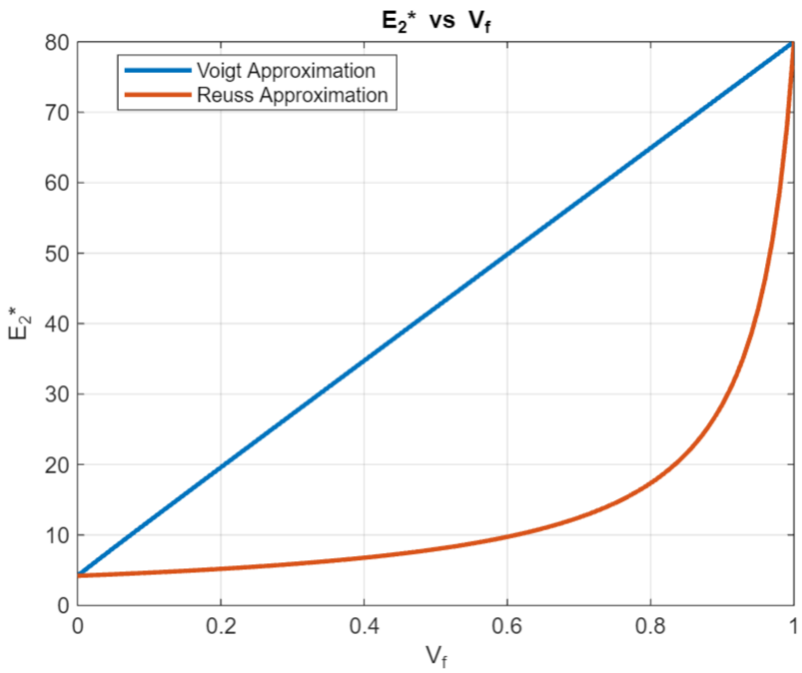
$$S_{ijkl} = S^{(m)}_{ijkl} + V_f \left(S^{(f)}_{ijrs} - S^{(m)}_{ijrs} \right)$$

Results:

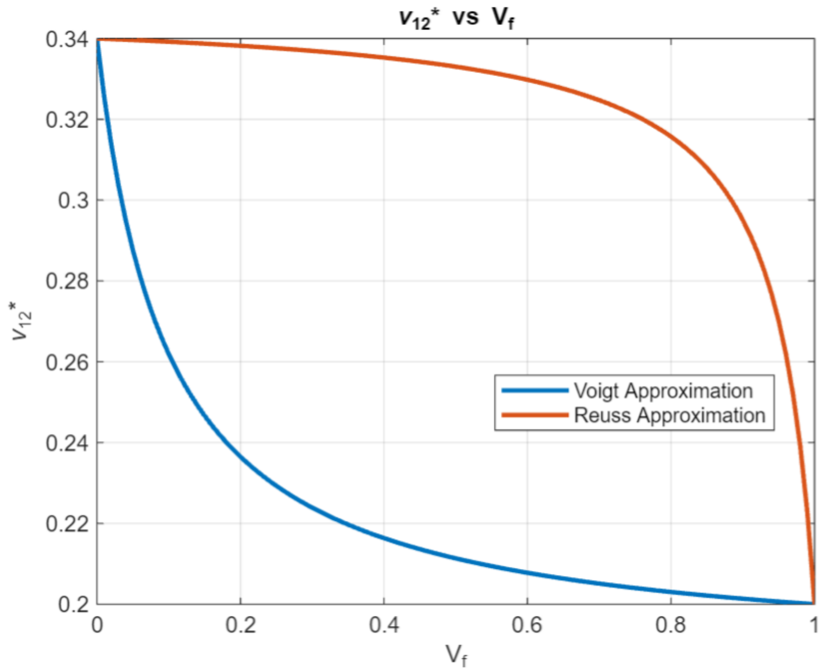
Effective Axial Modulus, E_1^* :



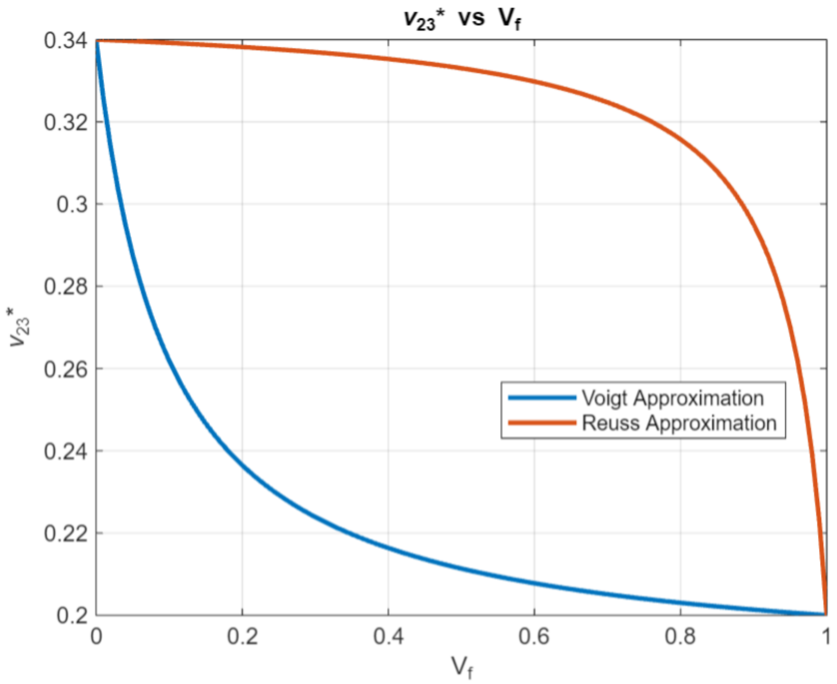
Effective Transverse Modulus, E_2^* :



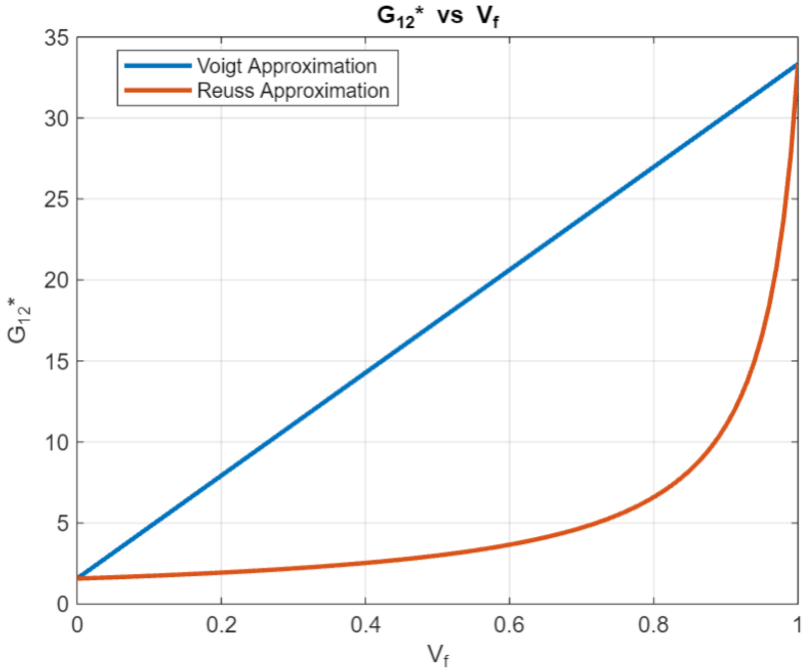
Effective Axial (Major) Poison's Ratio, v_{12}^{*} :



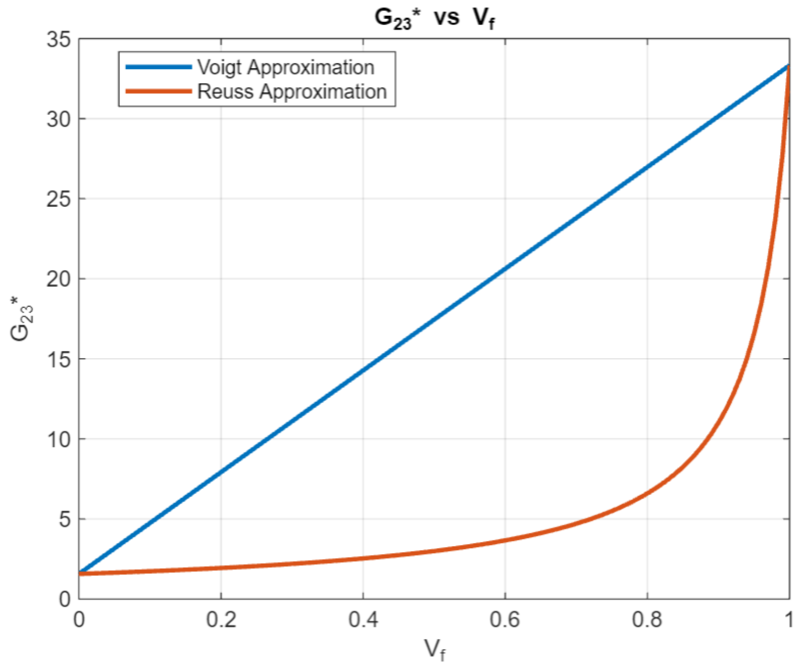
Effective Transverse Poison's Ratio, v_{23}^{*} :



Effective Axial Shear Modulus, G_{12}^* :



Effective Transverse Shear Modulus, G_{23}^* :



Concentric Cylinder Assemblage Model

To relate five effective independent stiffness coefficients with the measurable effective engineering constants, we typically focus on orthotropic or transversely isotropic materials. These effective engineering constants include:

$$E_1^*, K_{23}^*, G_{23}^*, v_{12}^*, G_{12}^*$$

We find them by the following formulas,

Effective Axial Modulus, E_1^* :

$$E_1^* = V_f E^{(f)} + \left(1 - V_f\right)E^{(m)} + \frac{4V_f\left(1 - V_f\right)\left(v_f - v_m\right)^2 \mu_m}{\dfrac{\left(1 - V_f\right)\mu_m}{\left(k_f + \dfrac{\mu_m}{2}\right)} + \dfrac{V_f \mu_m}{\left(k_m + \dfrac{\mu_m}{3}\right)} + 1}$$

Bulk Modulus K_{23}^* :

$$K_{23}^* = K_m + \frac{\mu_m}{3} + \frac{V_f}{k_f - k_m + \dfrac{1}{3}\left(\mu_f - \mu_m\right) + \dfrac{\left(1 - V_f\right)}{4\left(k_m + \dfrac{4}{3}\mu_m\right)}}$$

Effective Transverse Shear Modulus, G_{23}^* :

$$\frac{G_{23}^*}{G_m} = 1 + \frac{V_f}{\dfrac{G_m}{\left(G_{23}^{(f)} - G_m\right)} + \dfrac{\left(k_m + \dfrac{7}{3}G_m\right)}{\left(2k_m + \dfrac{8}{3}G_m\right)}}$$

Effective Axial Shear Modulus, G_{12}^* :

$$\frac{G_{12}^*}{G^{(m)}} = \frac{G_{12}^{(f)}\left(1 + V_f\right) + G^{(m)}\left(1 - V_f\right)}{G_{12}^{(f)}\left(1 - V_f\right) + G^{(m)}\left(1 + V_f\right)}$$

Axial (Major) Poison's Ratio, v_{12}^{*} :

$$v_{12}^{*}=\left(1-V_f\right)v^{(m)}+V_f v^{(f)}+\frac{V_f\left(1-V_f\right)\left(v^{(f)}-v^{(m)}\right)\left[\frac{\mu_m}{\left(k_m+\frac{\mu_m}{3}\right)}-\frac{\mu_m}{\left(k_f+\frac{\mu_f}{3}\right)}\right]}{\frac{\left(1-V_f\right)\mu_m}{\left(k_f+\frac{\mu_f}{3}\right)}+\frac{V_f\mu_m}{\left(k_m+\frac{\mu_m}{3}\right)}+1}$$

After finding the above 5 independent constants we find next 3 dependent constants:

$$E_2^{*}, v_{23}^{*}, v_{21}^{*}$$

We find them by the following formulas,

Effective Transverse Modulus, E_2^{*} :

$$E_2^{*}=\frac{4G_{23}^{*}K_{23}^{*}}{K_{23}^{*}+G_{23}^{*}+\frac{4\left(v_{12}^{*}\right)^2 G_{23}^{*}K_{23}^{*}}{E_1^{*}}}$$

Transverse Poison's Ratio, v_{23}^{*} :

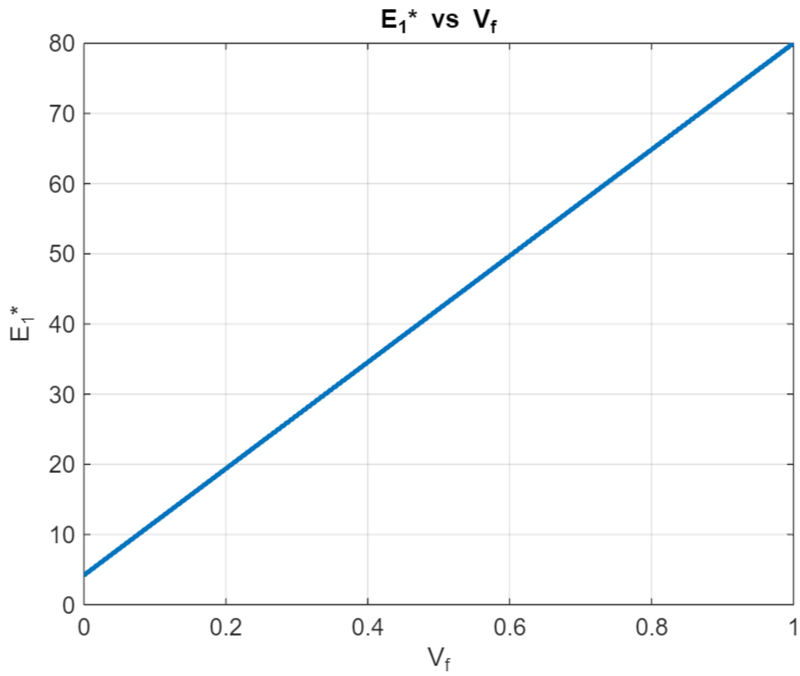
$$v_{23}^{*}=\frac{K_{23}^{*}-G_{23}^{*}-\frac{4\left(v_{12}^{*}\right)^2 G_{23}^{*}K_{23}^{*}}{E_1^{*}}}{K_{23}^{*}+G_{23}^{*}+4(v_{12}^{*})^2 G_{23}^{*}K_{23}^{*}}$$

Axial Poison's Ratio, v_{21}^{*}:

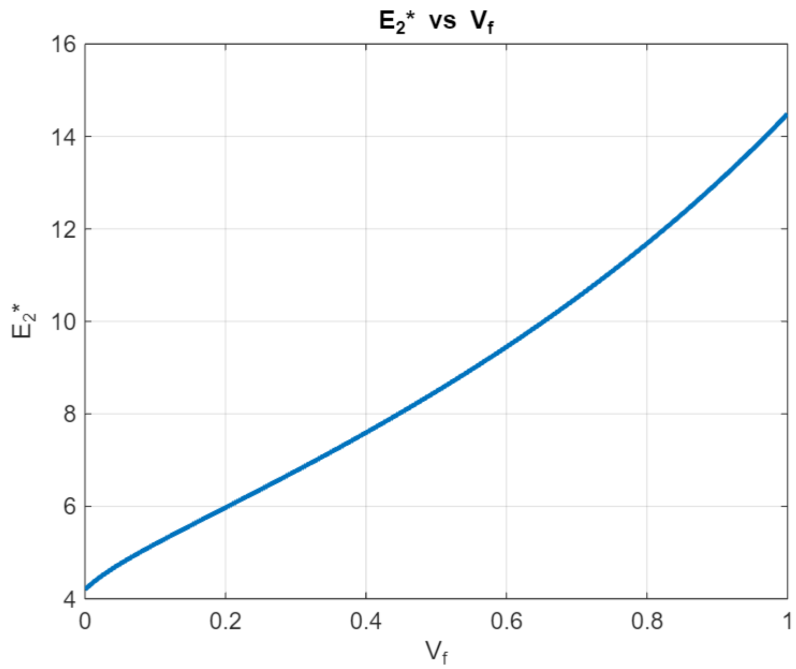
$$v_{21}^{*}=\frac{4\left(v_{12}^{*}\right)^2 G_{23}^{*}K_{23}^{*}}{E_1^{*}\left(K_{23}^{*}+G_{23}^{*}\right)+4\left(v_{12}^{*}\right)^2 G_{23}^{*}K_{23}^{*}}$$

Results:

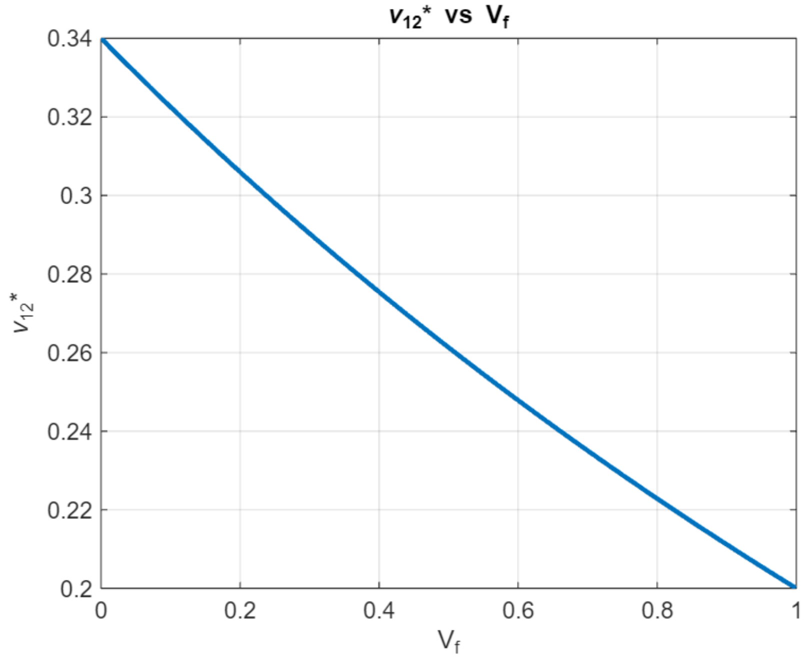
Effective Axial Modulus, E_1^* :



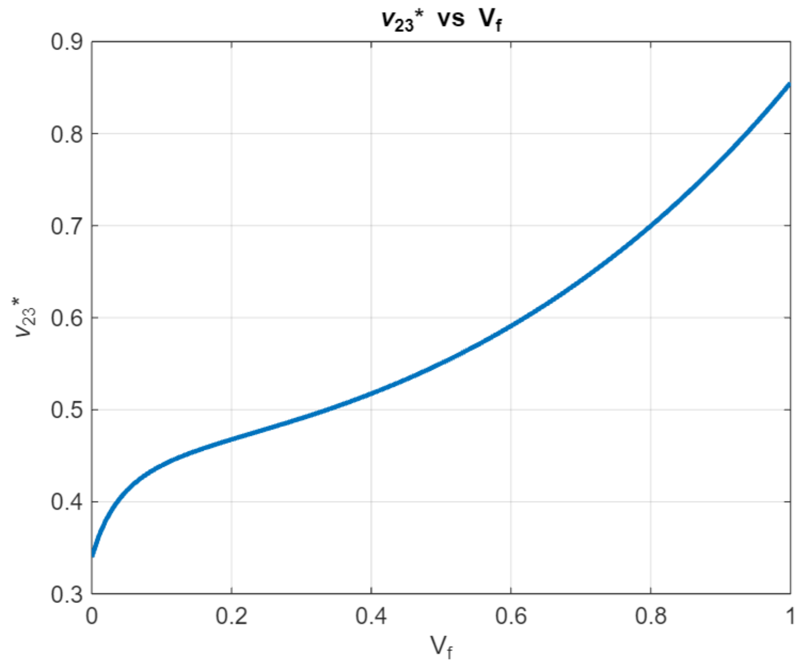
Effective Transverse Modulus, E_2^* :



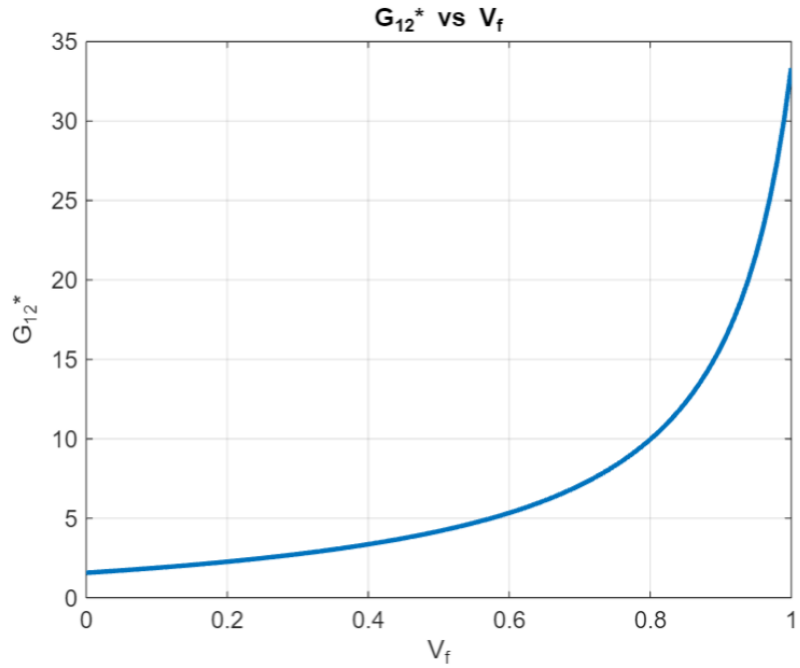
Effective Axial (Major) Poison's Ratio, $v_{12}^{(*)}$:



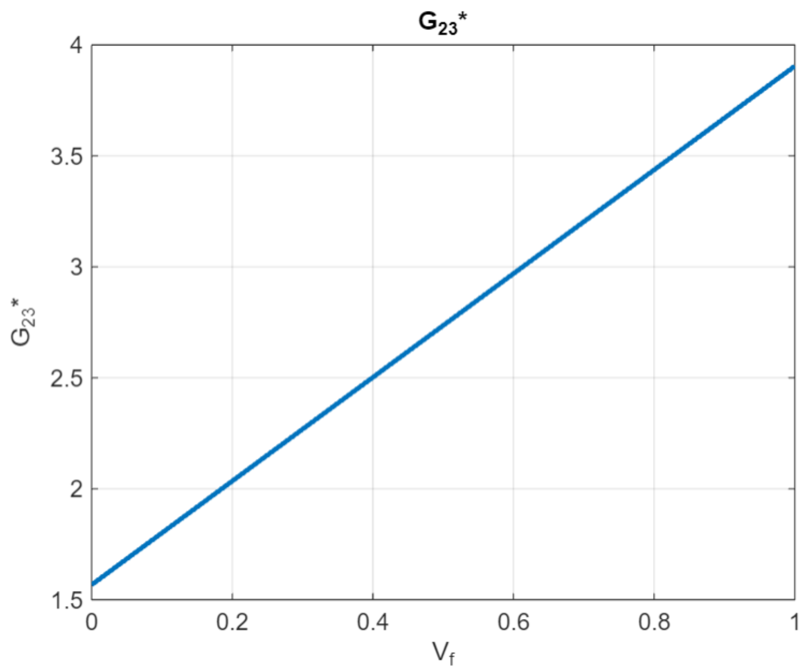
Effective Transverse Poison's Ratio, $v_{23}^{(*)}$:



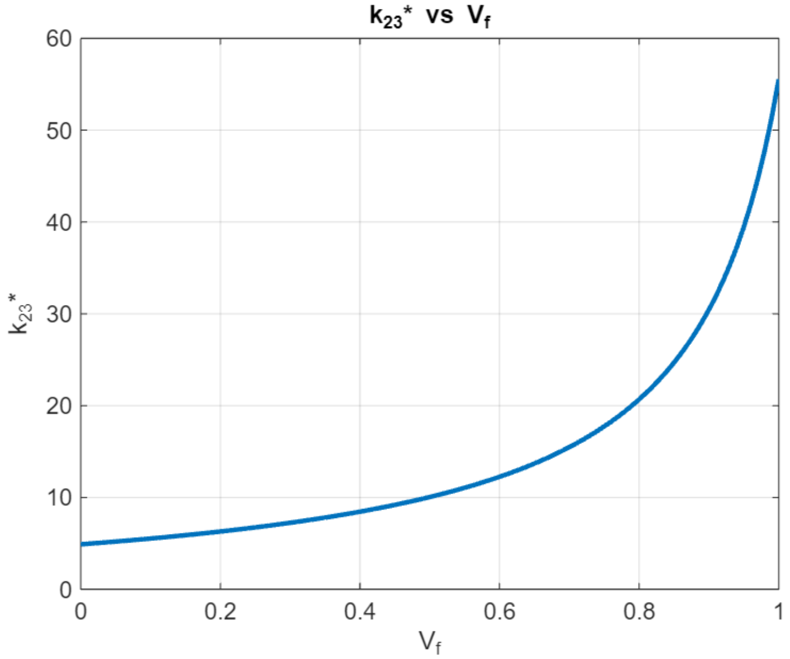
Effective Axial Shear Modulus, G_{12}^{*} :



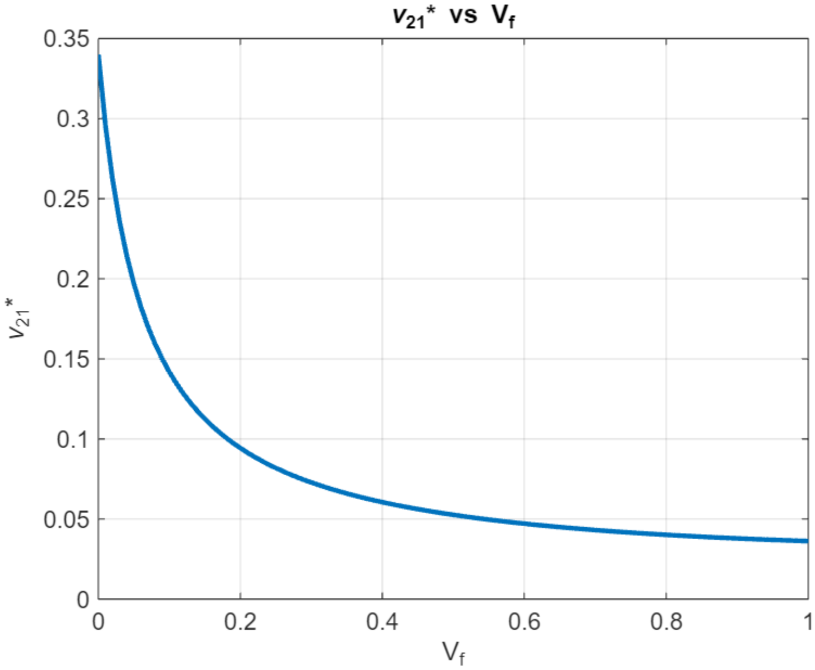
Effective Transverse Shear Modulus, G_{23}^{*} :



Effective Bulk Modulus K_{23}^{*} :



Effective Axial Poison's Ratio, $v_{21}^{(*)}$:



Self-consistent Method

Here we introduce the concept of hills moduli: *k, l, m, n, p*

We can relate these moduli to the compliance matrix with the formula below,

$$\begin{vmatrix} n & l & l & 0 & 0 & 0 \\ l & k+m & k-m & 0 & 0 & 0 \\ l & k-m & k+m & 0 & 0 & 0 \\ 0 & 0 & 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 \\ 0 & 0 & 0 & 0 & 0 & p \end{vmatrix}$$

After finding all the hills moduli for both the fibre and the matrix we can use the following formulas to solve for, k^*, m^*, p^*

$$\frac{V_f k_f}{k_f + m^*} + \frac{V_m k_m}{k_m + m^*} = 2\left[\frac{V_f m_m}{m_m - m^*} + \frac{V_m m_f}{m_f - m^*}\right]$$

$$\frac{1}{2p^*} = \frac{V_f}{p^* - p_m} + \frac{V_m}{p^* - p_f}$$

$$\frac{1}{k^* + m^*} = \frac{V_f}{k_f + m^*} + \frac{V_m}{k_m + m^*}$$

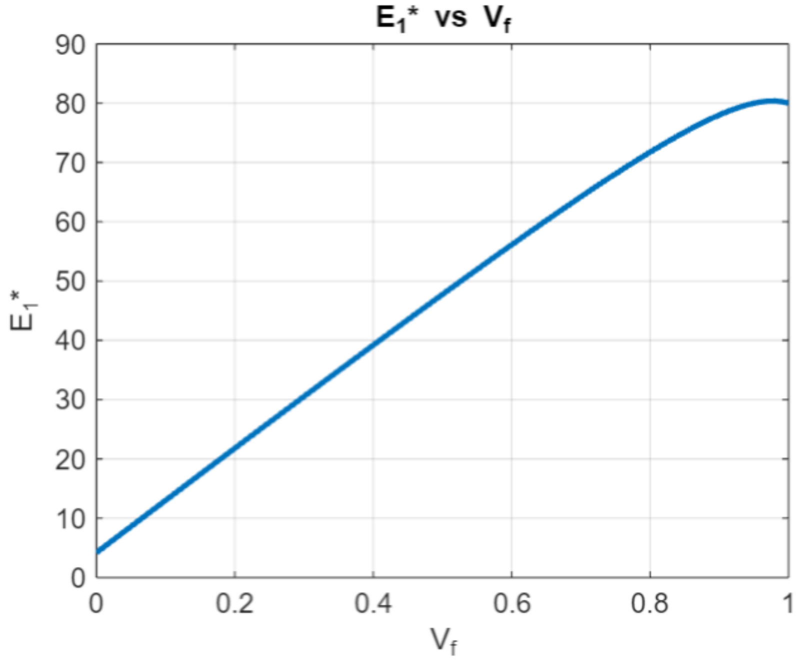
Note: To solve them and get an unique solution we assumed that the all the values are real and positive.

We then use the following relation to find the other two moduli, l^*, n^*

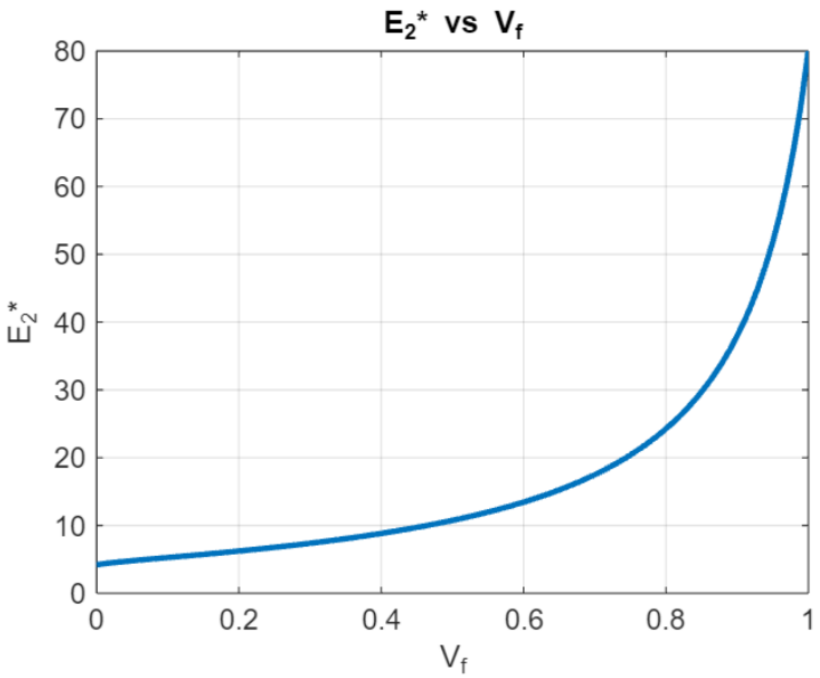
$$\frac{k^* - k_f}{l^* - l_f} = \frac{k^* - k_m}{l^* - l_m} = \frac{l^* - V_f l_f - V_m l_m}{n^* - V_f n_f - V_m n_m} = \frac{k_f - k_m}{l_f - l_m}$$

Results:

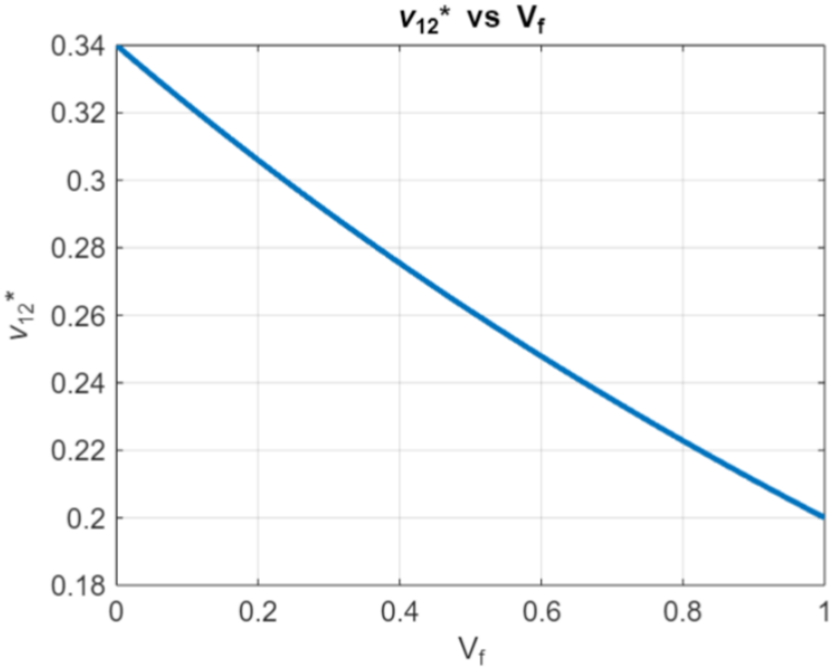
Effective Axial Modulus, E_1^* :



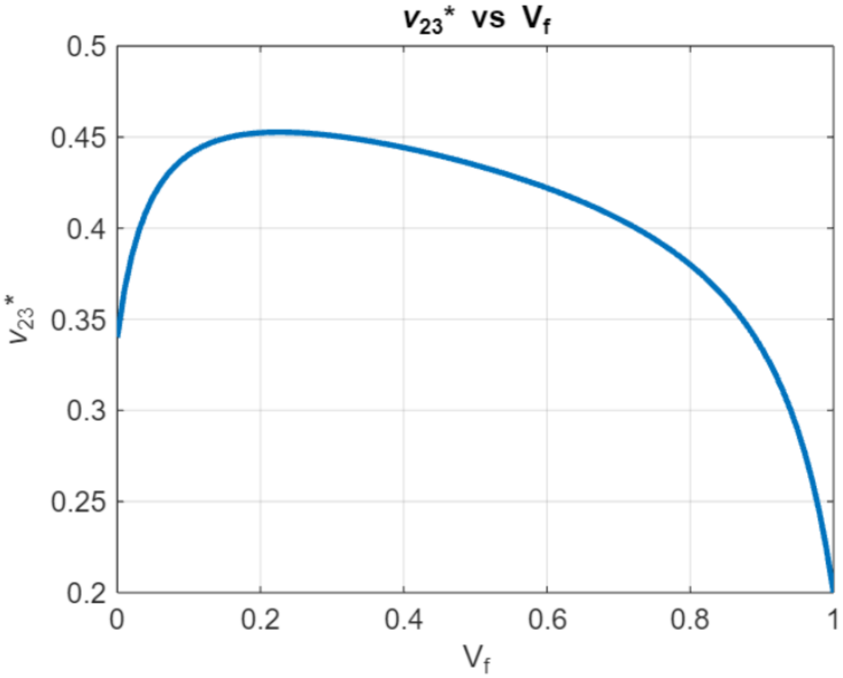
Effective Transverse Modulus, E_2^* :



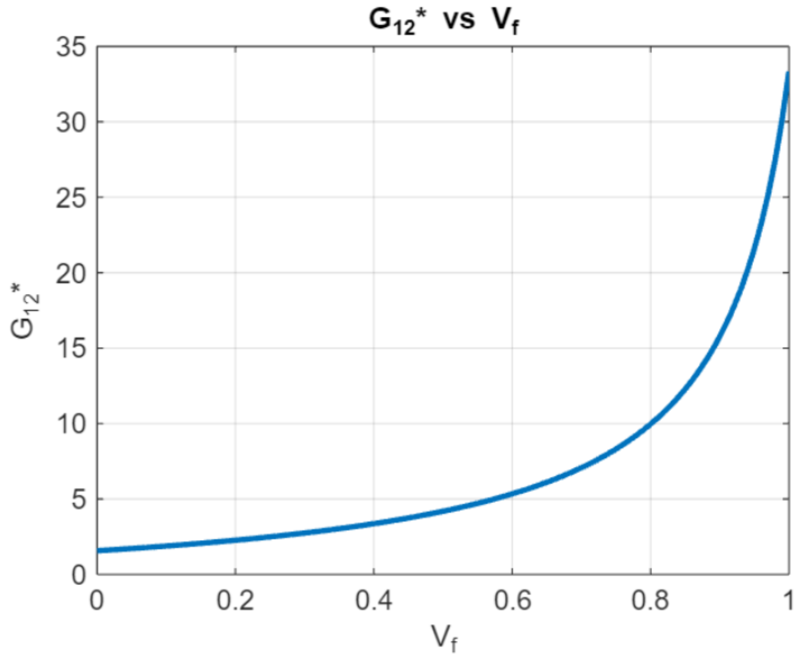
Effective Axial (Major) Poison's Ratio, v_{12}^{*} :



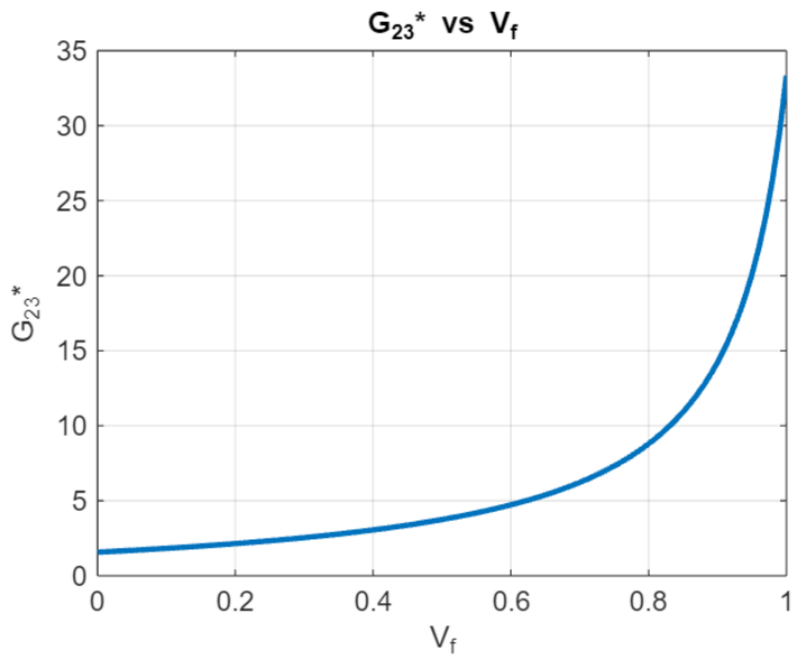
Effective Transverse Poison's Ratio, v_{23}^{*} :



Effective Axial Shear Modulus, G_{12}^{*} :



Effective Transverse Shear Modulus, G_{23}^{*}



Mori-Tanaka Method

The Mori-Tanaka method provides direct formulas for estimating the effective properties of composite materials. It accounts for the interaction between the matrix and the inclusions by considering the inclusions to be embedded in an infinite matrix subjected to the matrix's average stress. Dvorak et al. have presented explicit relations in terms of Hill's moduli, which offer improved accuracy for predicting the behaviour of composites.

Dvorak et al have given the explicit relations in terms of Hill's moduli as,

$$k = \frac{V_f k_f (k_m + m_m) + V_m k_m (k_f + m_m)}{V_f (k_m + m_m) + V_m (k_f + m_m)}$$

$$l = \frac{V_f l_f (k_m + m_m) + V_m l_m (k_f + m_m)}{V_f (k_m + m_m) + V_m (k_f + m_m)}$$

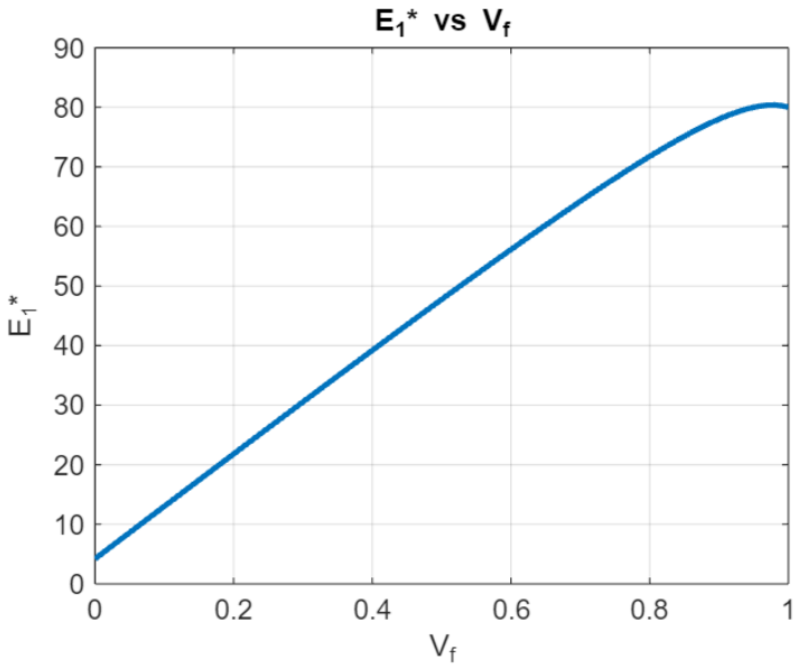
$$n = V_m n_m + V_f n_f + (1 - V_f l_f - V_m l_m)\left(\frac{l_f - l_m}{k_f - m_m}\right)$$

$$m = \frac{m_f m_m (k_m + 2m_m) + k_m m_m (V_f m_f + V_m m_m)}{k_m m_m + (k_m + 2m_m)(V_f m_m + V_m m_f)}$$

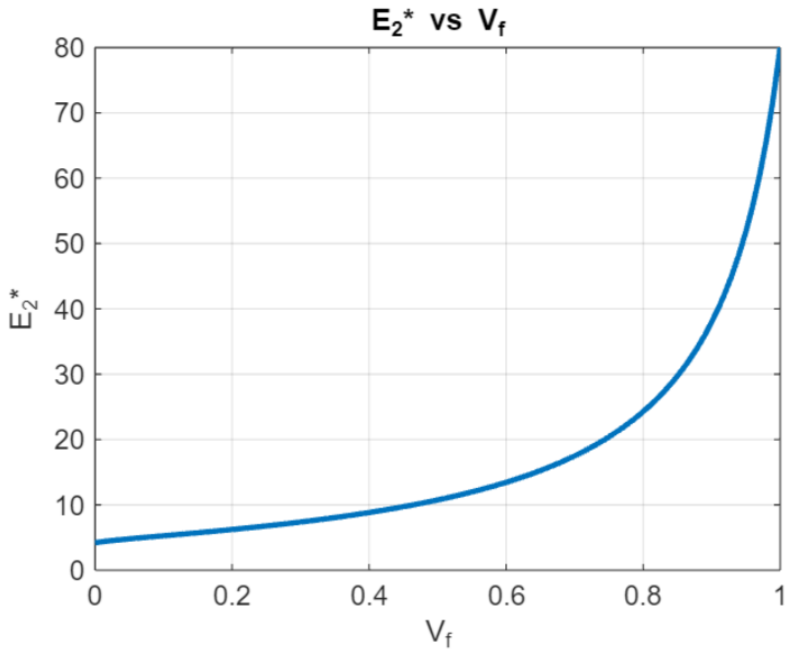
$$P = \frac{2V_f \rho_m \rho_f + V_m (\rho_m \rho_f + \rho_m^2)}{2V_f \rho_m + V_m (\rho_f + \rho_m)}$$

Result:

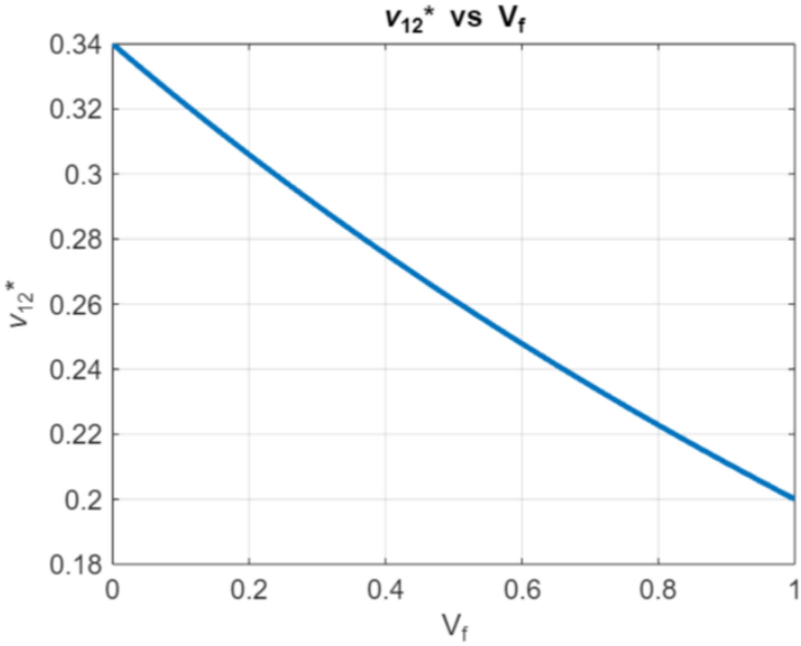
Effective Axial Modulus, E_1^* :



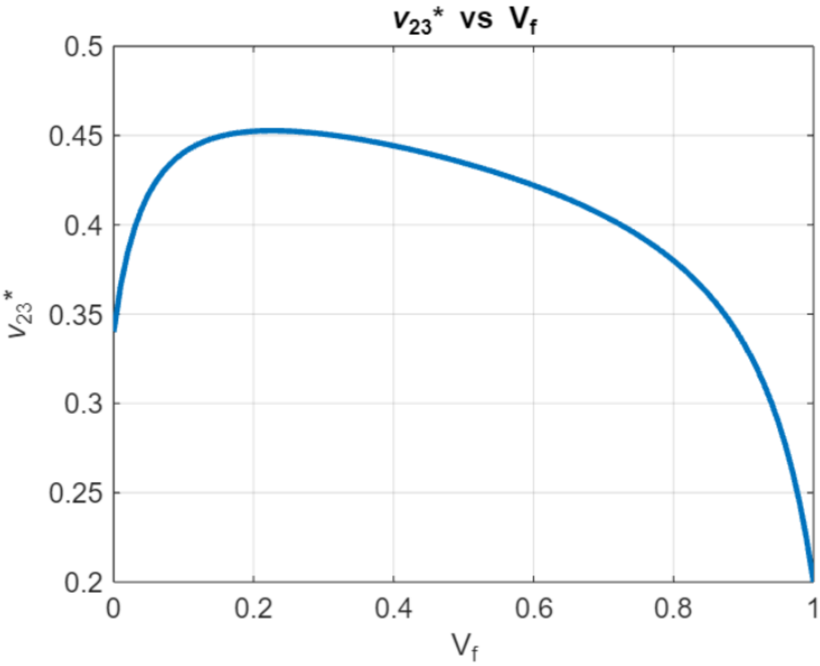
Effective Transverse Modulus, E_2^* :



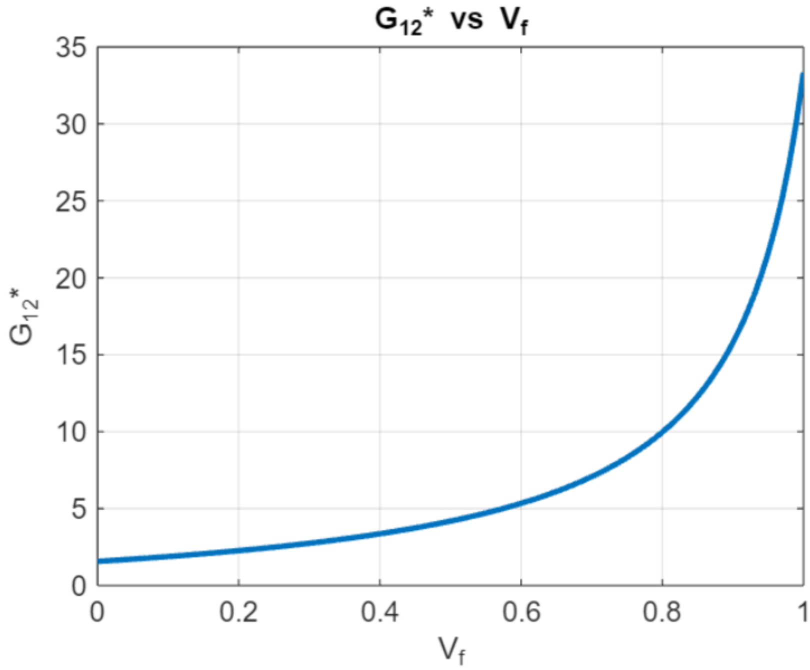
Effective Axial (Major) Poison's Ratio, v_{12}^{*} :



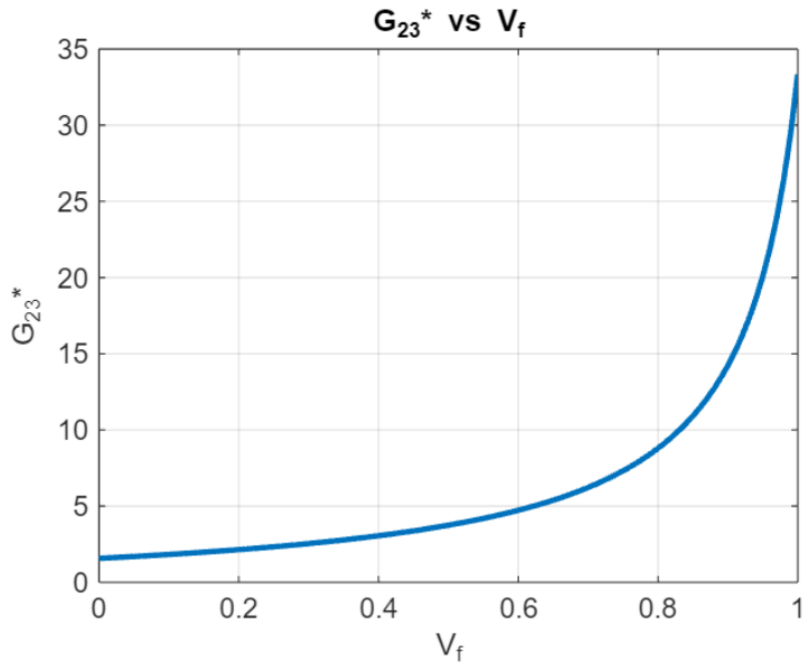
Effective Transverse Poison's Ratio, v_{23}^{*} :



Effective Axial Shear Modulus, G_{12}^{*} :



Effective Transverse Shear Modulus, G_{23}^{*} :



Halpin-Tsai method

The longitudinal Young's modulus is same as given by rule of mixtures using strength of materials approach.

$$E_1^* = E_1^{(f)} V_f + E_1^{(m)} V_m$$

The axial Poisson's ratio is same as given by rule of mixtures using strength of materials approach.

$$v_{12}^* = v_{12}^{(f)} V_f + v_{12}^{(m)} V_m$$

Then the equation to solve for other constants is given below,

$$\frac{M^*}{M^{(m)}} = \frac{1 + \zeta \eta V_f}{1 - \eta V_f}$$

$$\eta = \frac{\left[\frac{M^{(f)}}{M^{(m)}} - 1\right]}{\left[\frac{M^{(f)}}{M^{(m)}} + \zeta\right]}$$

Here M represents E_2, G_{12}, v_{23}

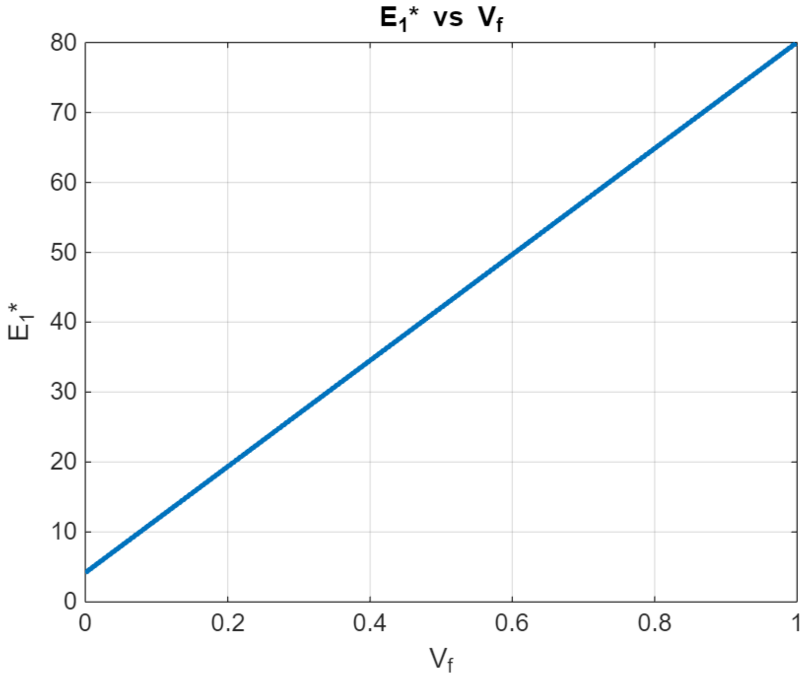
The remaining constant G23 is given by,

$$\frac{E_2^*}{2(1 + v_{23}^*)}$$

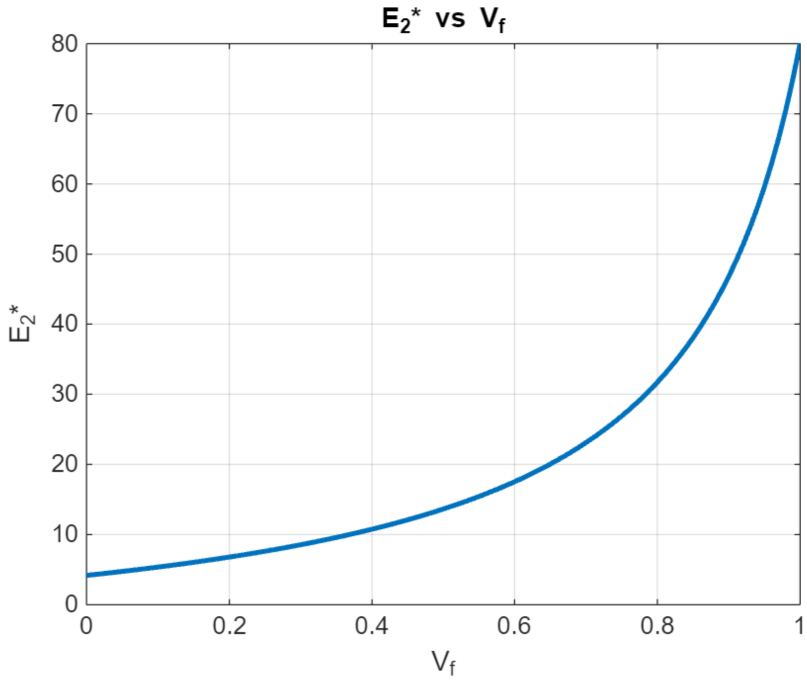
Note: In the Halpin-Tsai method, the parameter ζ (zeta) is a geometry-dependent empirical constant that accounts for the shape and orientation of the reinforcing fibres. For circular fibres aligned in the longitudinal direction, ζ is typically taken as 2. This choice reflects the influence of fibre geometry on the reinforcement efficiency and provides accurate predictions of the composite's mechanical properties. The assumption of ζ = 2 is widely accepted in literature for unidirectional composites with continuous circular fibres.

Results:

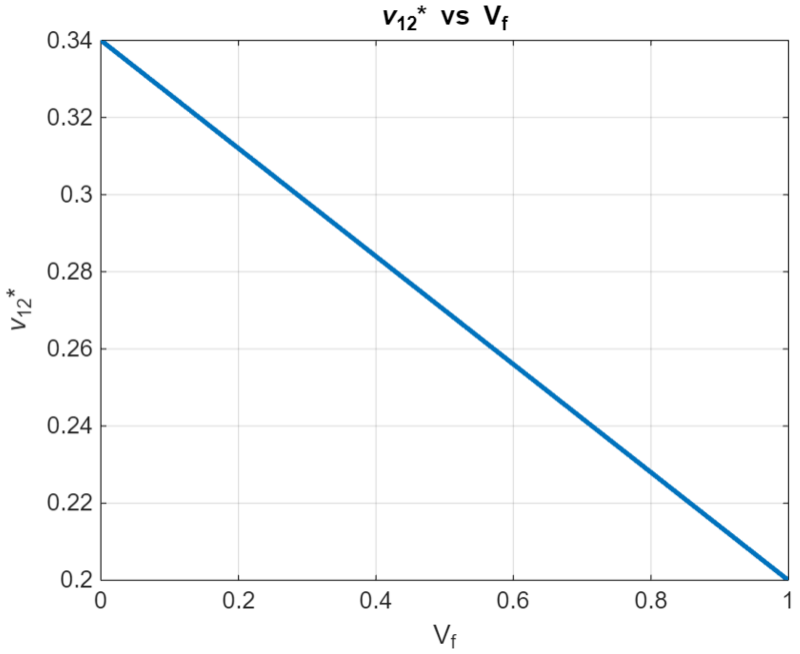
Effective Axial Modulus, E_1^* :



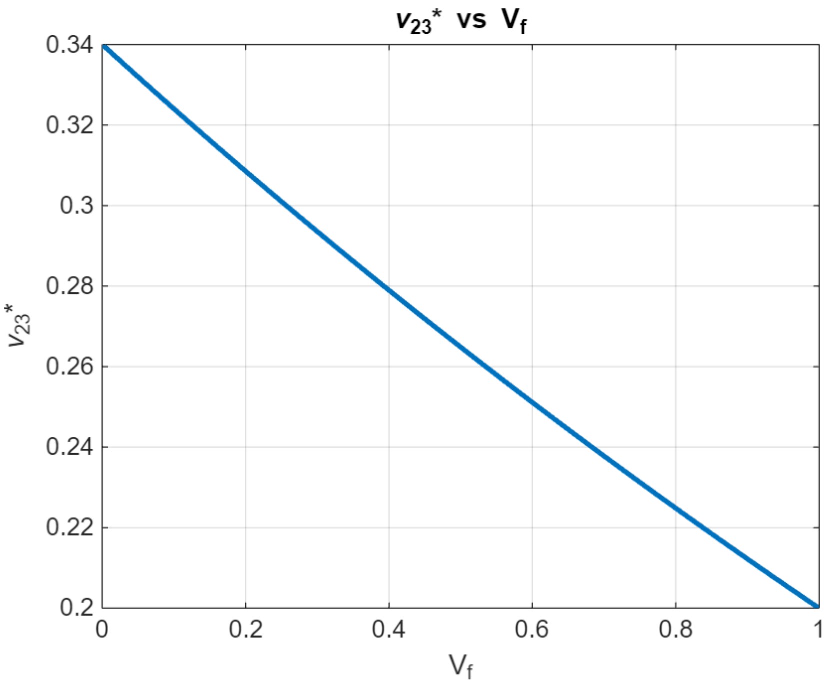
Effective Transverse Modulus, E_2^* :



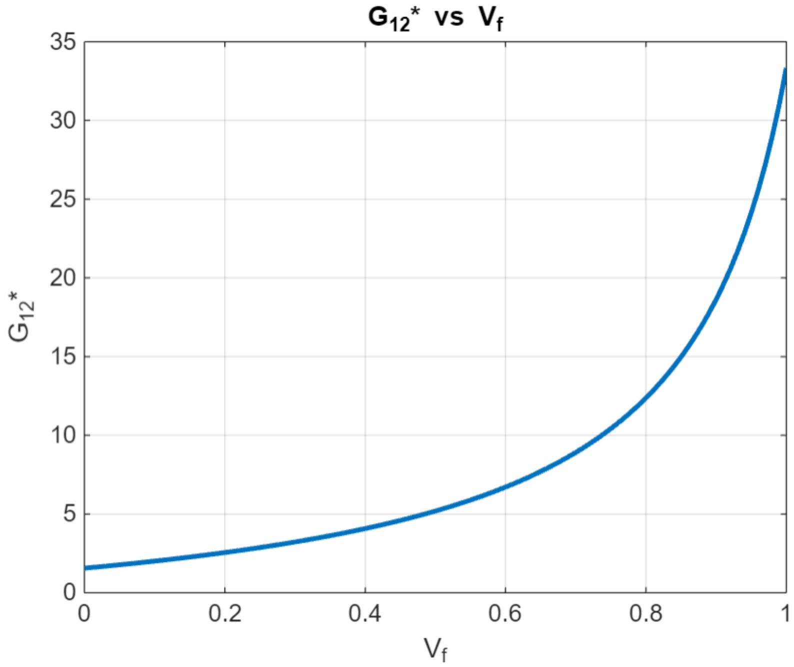
Effective Axial (Major) Poison's Ratio, v_{12}^* :



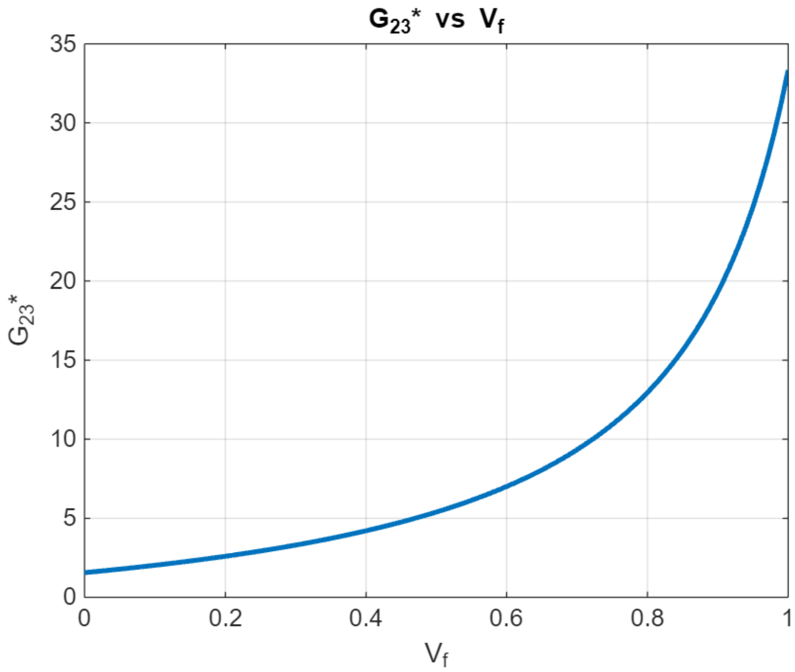
Transverse Poison's Ratio, v_{23}^* :



Effective Axial Shear Modulus, G_{12}^* :



Effective Transverse Shear Modulus, G_{23}^* :



Hashin Shtrikman Bounds

Hashin and Shtrikman have given an approach to find upper and lower limits on the engineering constants for a heterogeneous material using principles of minimum potential energy and minimum complementary energy. There are no assumptions involved regarding arrangement of fibres in the matrix and geometry of fibres. The following closed form expressions are given by Hashin.

Lower and upper limits on the engineering constants are:

Bulk modulus K:

$$K_{(-)} = K_m + \frac{V_f}{\left[\frac{1}{K_f - K_m} + \frac{V_m}{K_m + G_m}\right]}$$

$$K_{(+)} = K_f + \frac{V_m}{\left[\frac{1}{K_m - K_f} + \frac{V_f}{K_f + G_f}\right]}$$

Transverse Shear Modulus G_{23}:

$$G_{23}^{(-)} = G_{23}^{m} + \frac{V_f}{\left[\frac{1}{G_{23}^{f} - G_{23}^{m}} + \frac{(K_m + 2G_m)V_m}{2G_m(K_m + G_m)}\right]}$$

where,

$$\gamma = \frac{G_{23}^{f}}{G_{23}^{m}}$$

$$\beta_1 = \frac{K_f}{K_f + 2G_{23}^{m}}, \quad \beta_2 = \frac{K_f}{K_f + 2G_{23}^{f}}$$

$$\alpha = \frac{\beta_1 - \gamma\beta_2}{1 + \gamma\beta_2}, \quad \rho = \frac{\gamma + \beta_1}{\gamma - 1}$$

$$G_{23}^{(+)} = G_{23}^{m}\left[1 + \frac{(1+\beta_1)V_f}{\left[\rho - V_f\left(1 + \frac{3\beta_2^2 V_m^2}{\alpha V_f^3 + 1}\right)\right]}\right]$$

Axial Modulus E_1:

$$E_1^{(-)} = E_1^m V_m + E_1^f V_f + \frac{4\left(v_{12}^f - v_{12}^m\right)^2 V_f V_m}{\left[\frac{V_m}{K_f} + \frac{V_f}{K_m} + \frac{1}{G_{23}^m}\right]}$$

$$E_1^{(+)} = E_1^m V_m + E_1^f V_f + \frac{4\left(v_{12}^f - v_{12}^m\right)^2 V_f V_m}{\left[\frac{V_m}{K_f} + \frac{V_f}{K_m} + \frac{1}{G_{23}^f}\right]}$$

Major Poisson's ratio v_{12}:

$$v_{12}^{(-)} = v_{12}^m V_m + v_{12}^f V_f + \frac{\left(v_{12}^m - v_{12}^f\right)\left(\frac{1}{K_f} - \frac{1}{K_m}\right)V_f V_m}{\left[\frac{V_m}{K_f} + \frac{V_f}{K_m} + \frac{1}{G_{23}^f}\right]}$$

$$v_{12}^{(+)} = v_{12}^m V_m + v_{12}^f V_f + \frac{\left(v_{12}^f - v_{12}^m\right)\left(\frac{1}{K_m} - \frac{1}{K_f}\right)V_f V_m}{\left[\frac{V_m}{K_f} + \frac{V_f}{K_m} + \frac{1}{G_{23}^f}\right]}$$

Transverse Shear Modulus G_{12}:

$$G_{12}^{(-)} = \frac{G_{12}^m\, G_{12}^f\, V_m + G_{12}^f\left(1 + V_f\right)}{G_{12}^f\, V_m + G_{12}^m\left(1 + V_f\right)}$$

$$G_{12}^{(+)} = \frac{G_{12}^f G_{12}^f V_f + G_{12}^m(1 + V_m)}{G_{12}^m V_f + G_{12}^f(1 + V_m)}$$

Transverse Modulus E_2:

$$E_2^{(-)} = \left[\frac{4G_{23}^{(-)}K_{(-)}}{K_{(-)} + G_{23}^{(-)} + \left(\frac{4G_{23}^{(-)}K_{(-)}\left(v_{12}^{(+)}\right)^2}{E_1^{(-)}}\right)}\right]$$

$$E_2^{(+)} = \left[\frac{4G_{23}^{(+)}K_{(+)}}{K_{(+)} + G_{23}^{(+)} + \left(\frac{4G_{23}^{(+)}K_{(+)}\left(v_{12}^{(-)}\right)^2}{E_1^{(+)}}\right)}\right]$$

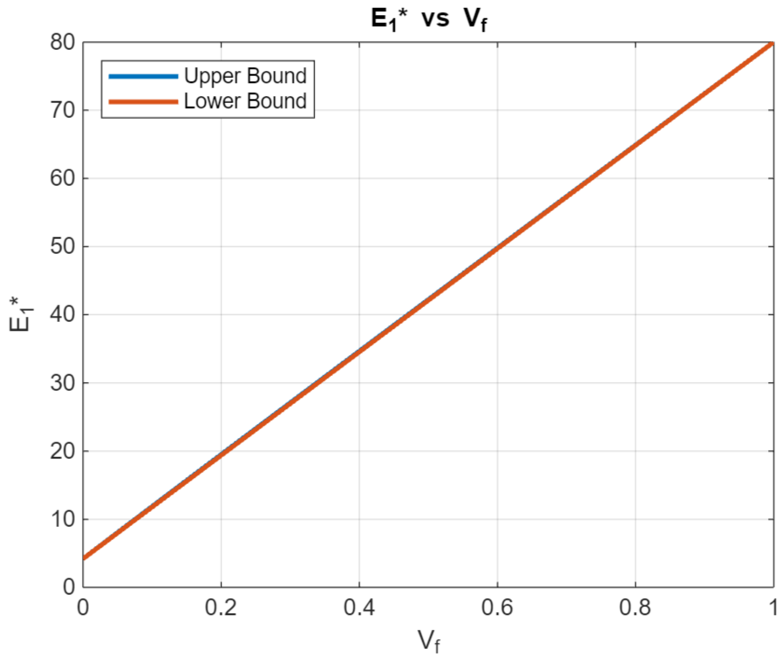
Major Poisson's ratio v_{23}:

$$v_{23}^{(-)} = \frac{\left(K_{(-)} - G_{23}^{(+)}\right) - \left(\frac{4G_{23}^{(+)}K_{(-)}\left(v_{12}^{(+)}\right)^2}{E_1^{(-)}}\right)}{K_{(-)} + G_{23}^{(+)} + \left(\frac{4G_{23}^{(+)}K_{(-)}\left(v_{12}^{(+)}\right)^2}{E_1^{(-)}}\right)}$$

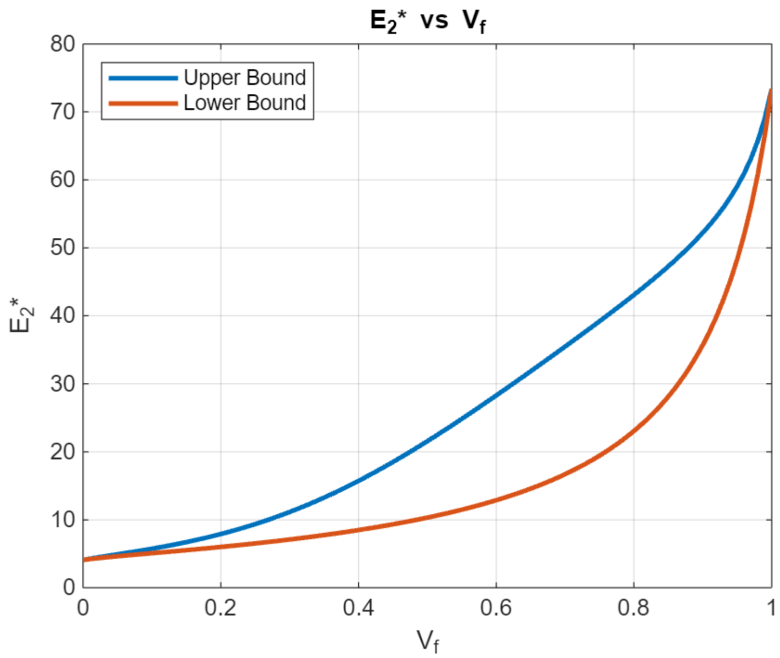
$$v_{23}^{(+)} = \frac{\left(K_{(+)} - G_{23}^{(-)}\right) - \left(\frac{4G_{23}^{(-)}K_{(+)}\left(v_{12}^{(-)}\right)^2}{E_1^{(+)}}\right)}{K_{(+)} + G_{23}^{(-)} + \left(\frac{4G_{23}^{(-)}K_{(+)}\left(v_{12}^{(-)}\right)^2}{E_1^{(+)}}\right)}$$

Results:

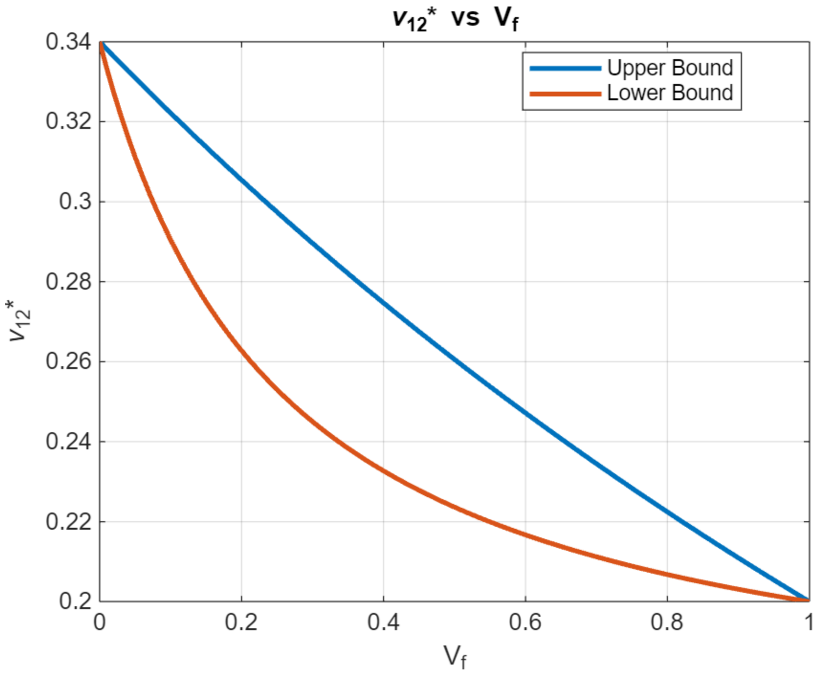
Effective Axial Modulus, E_1^* :



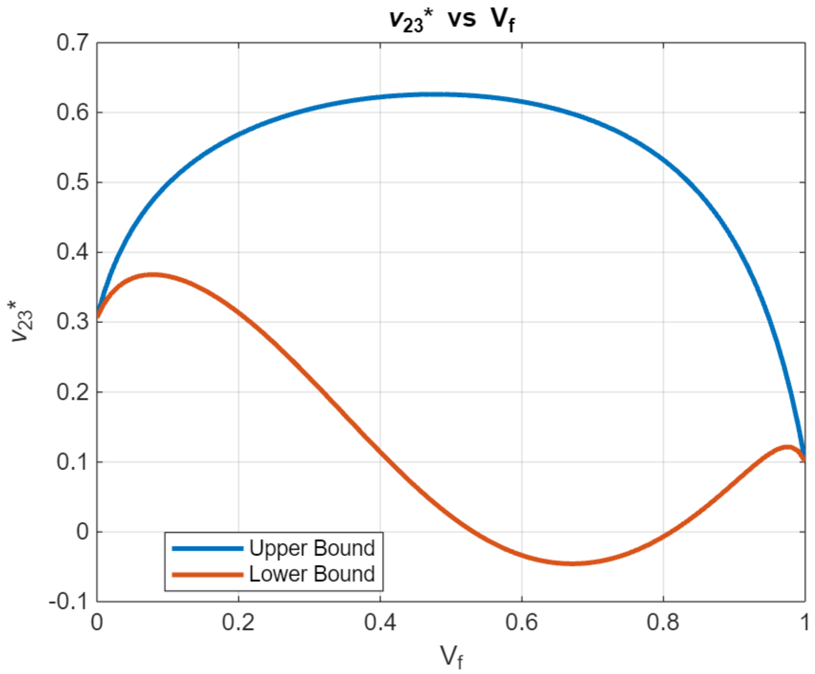
Effective Transverse Modulus, E_2^* :



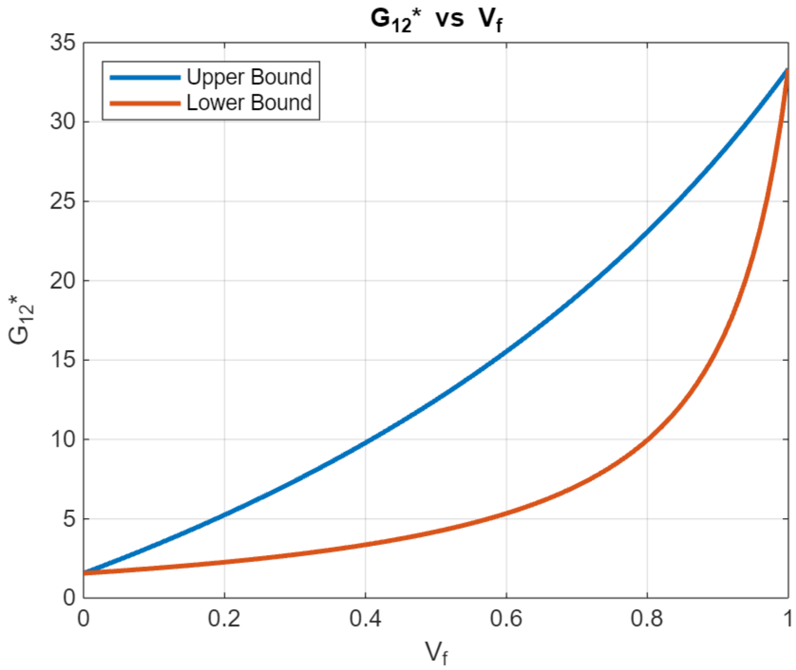
Effective Axial (Major) Poison's Ratio, v_{12}^{*} :



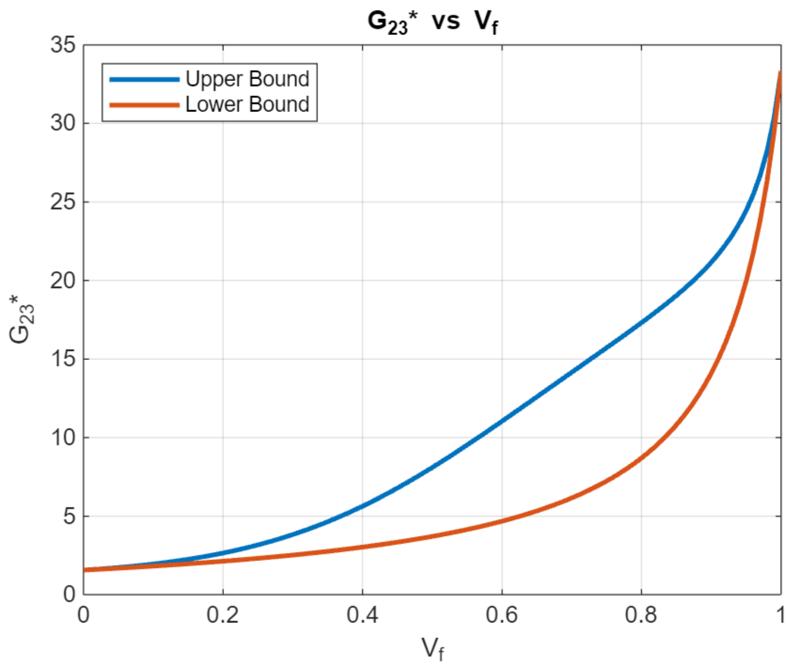
Effective Transverse Poison's Ratio, v_{23}^{*} :



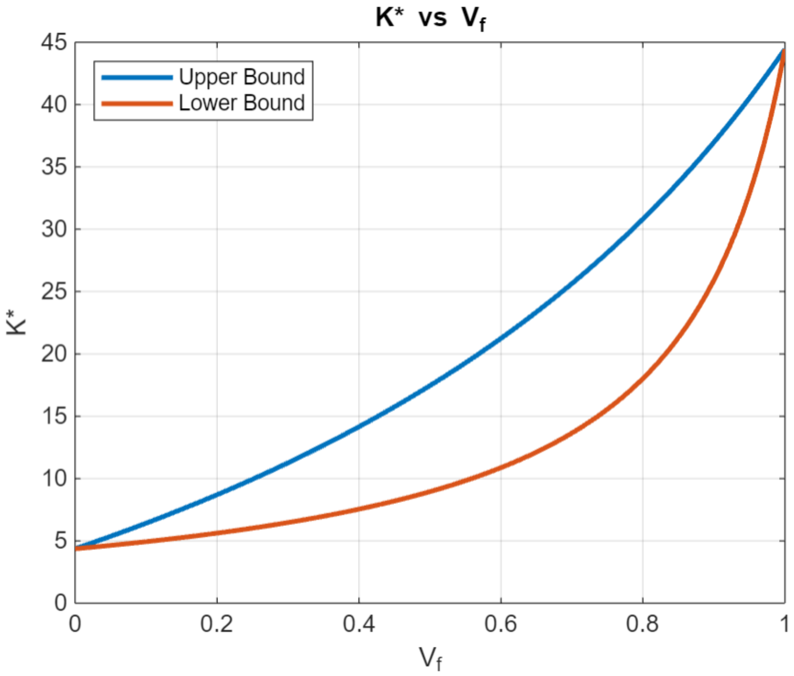
Effective Axial Shear Modulus, G_{12}^{*} :



Effective Transverse Shear Modulus, G_{23}^{*} :

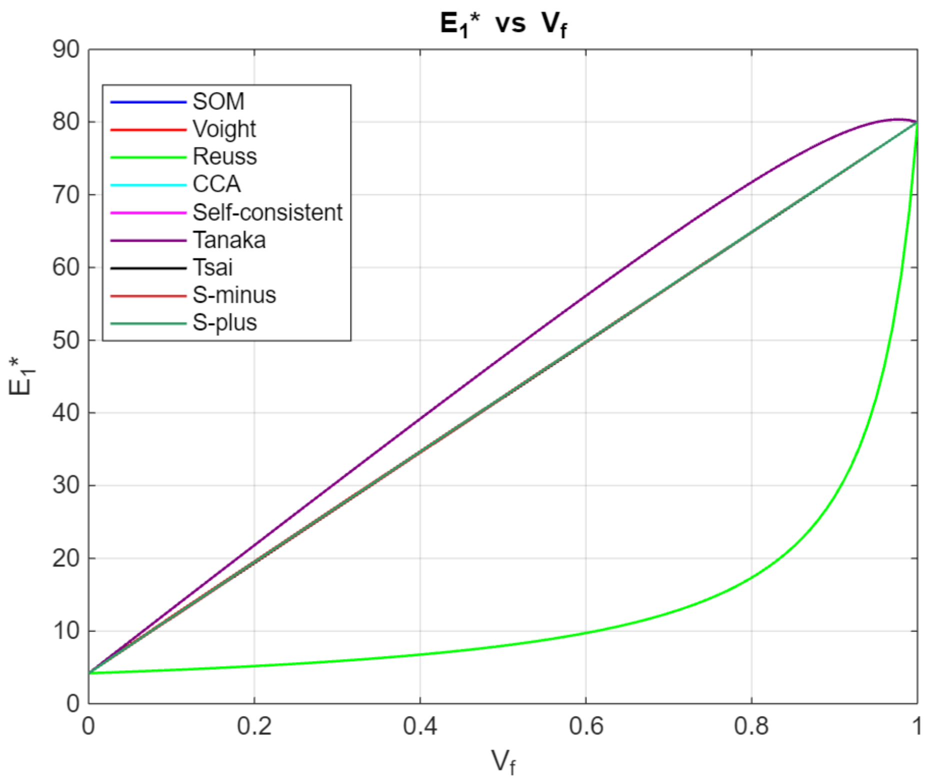


Effective Bulk Modulus K^* :

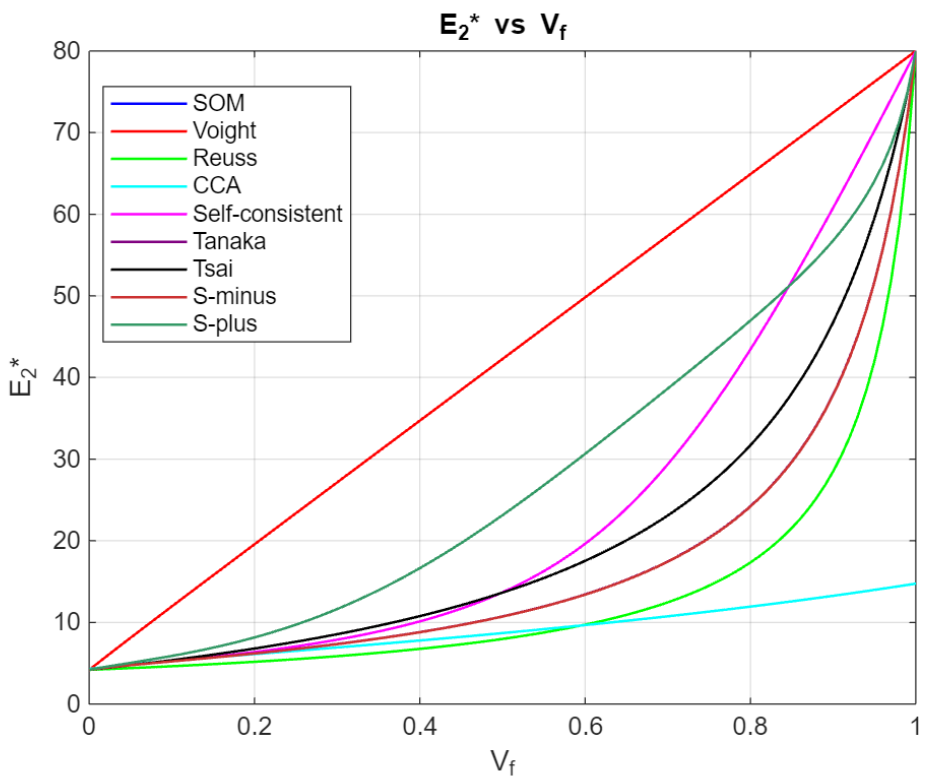


Results: (All Methods Together)

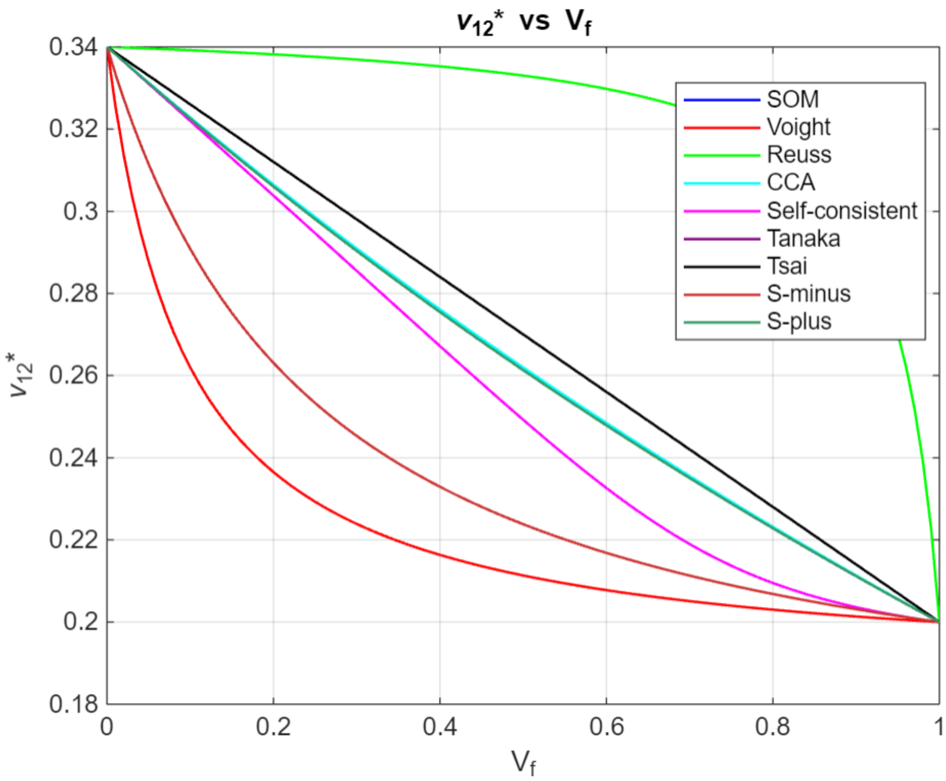
Effective Axial Modulus, E_1^* :



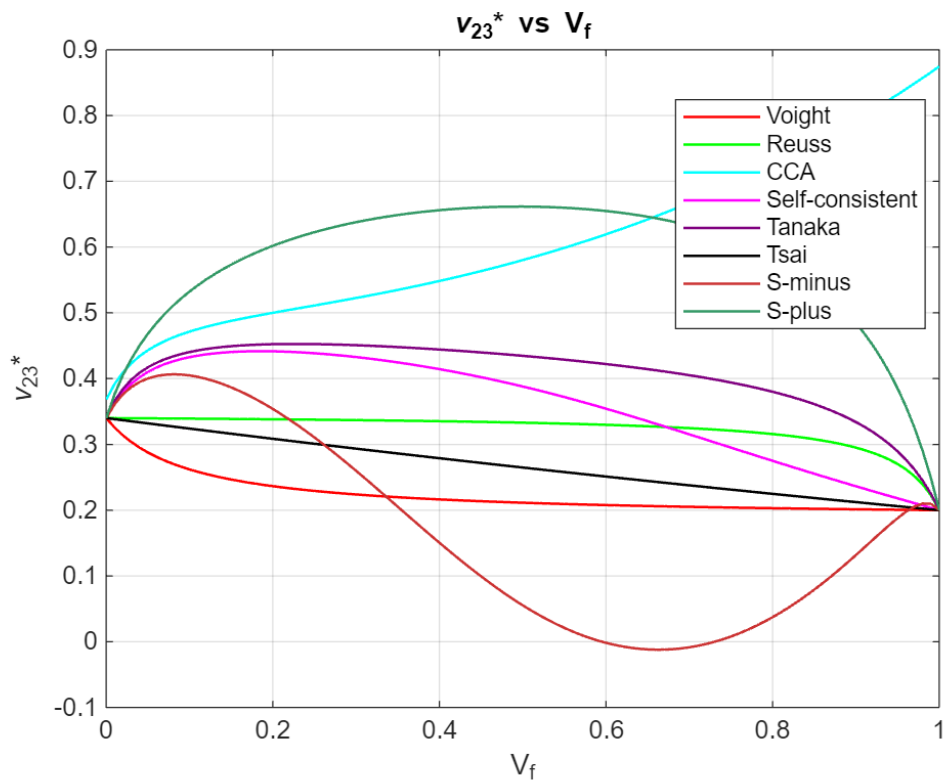
Effective Transverse Modulus, E_2^* :



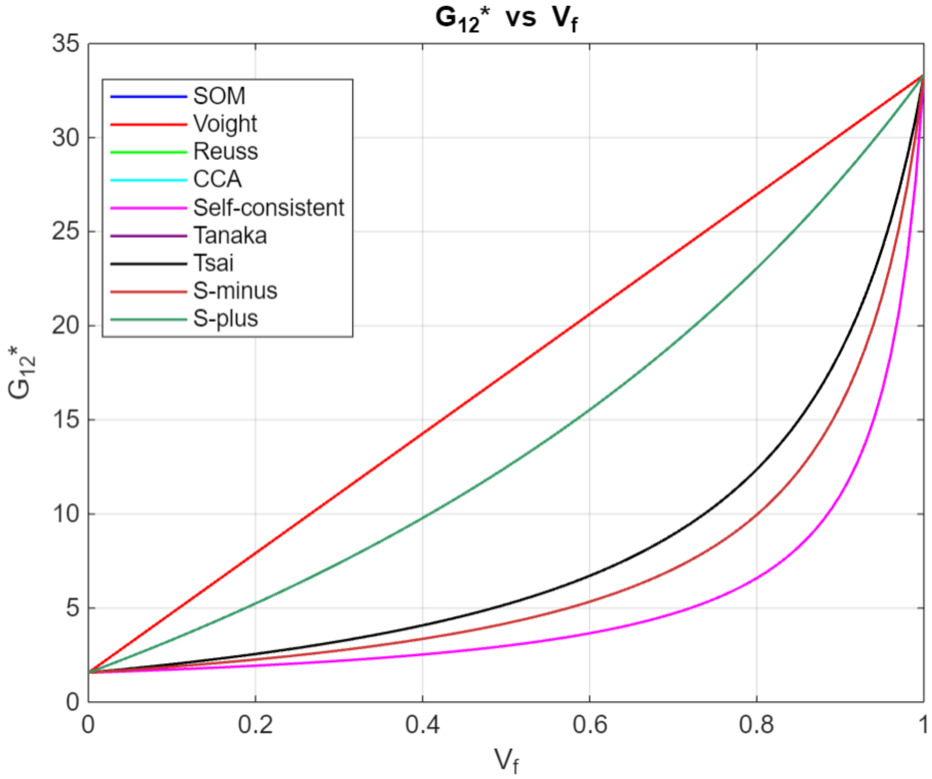
Effective Axial (Major) Poison's Ratio, v_{12}^{*} :



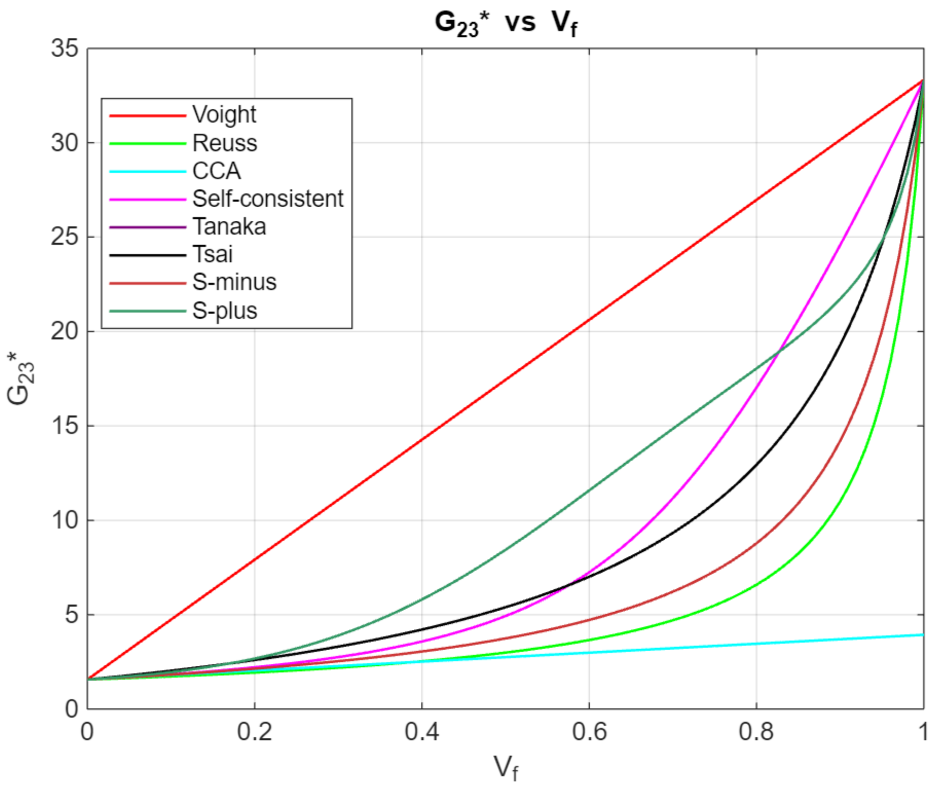
Effective Transverse Poison's Ratio, v_{23}^{*} :



Effective Axial Shear Modulus, G_{12}^{*} :



Effective Transverse Shear Modulus, G_{23}^{*} :



Conclusion

When both the fibre and the matrix materials are isotropic, and the fibres are aligned in a single direction within the matrix (i.e., a unidirectional composite), the resulting composite material exhibits transverse isotropy.

APPENDIX

(ALL CODES)

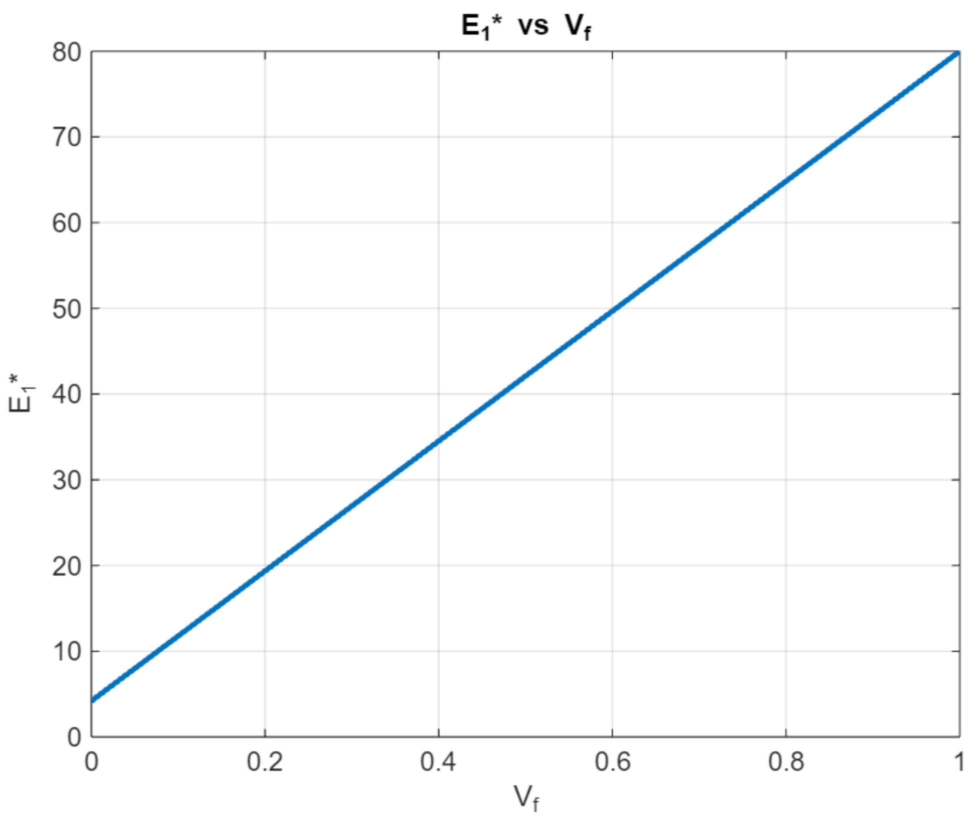
Strength of Materials Approach

```
clc;clear;
syms Vf
Vf_val = linspace(0,1,101);

%fiber
E_f = 80;
G_f = 33.33;
v_f = 0.2;
a_f = 4.9 * 1e-6;

S_f = [1/E_f, -v_f/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, 1/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, -v_f/E_f, 1/E_f, 0, 0, 0;
       0, 0, 0, 1/G_f, 0, 0;
       0, 0, 0, 0, 1/G_f, 0;
       0, 0, 0, 0, 0, 1/G_f;];
C_f = inv(S_f);

%matrix
E_m = 4.2;
v_m = 0.34;
a_m = 45 * 1e-6;
G_m = E_m/(2*(1+v_m));

S_m = [1/E_m, -v_m/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, 1/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, -v_m/E_m, 1/E_m, 0, 0, 0;
       0, 0, 0, 1/G_m, 0, 0;
       0, 0, 0, 0, 1/G_m, 0;
       0, 0, 0, 0, 0, 1/G_m;];
C_m = inv(S_m);
```

Strength of materials approach

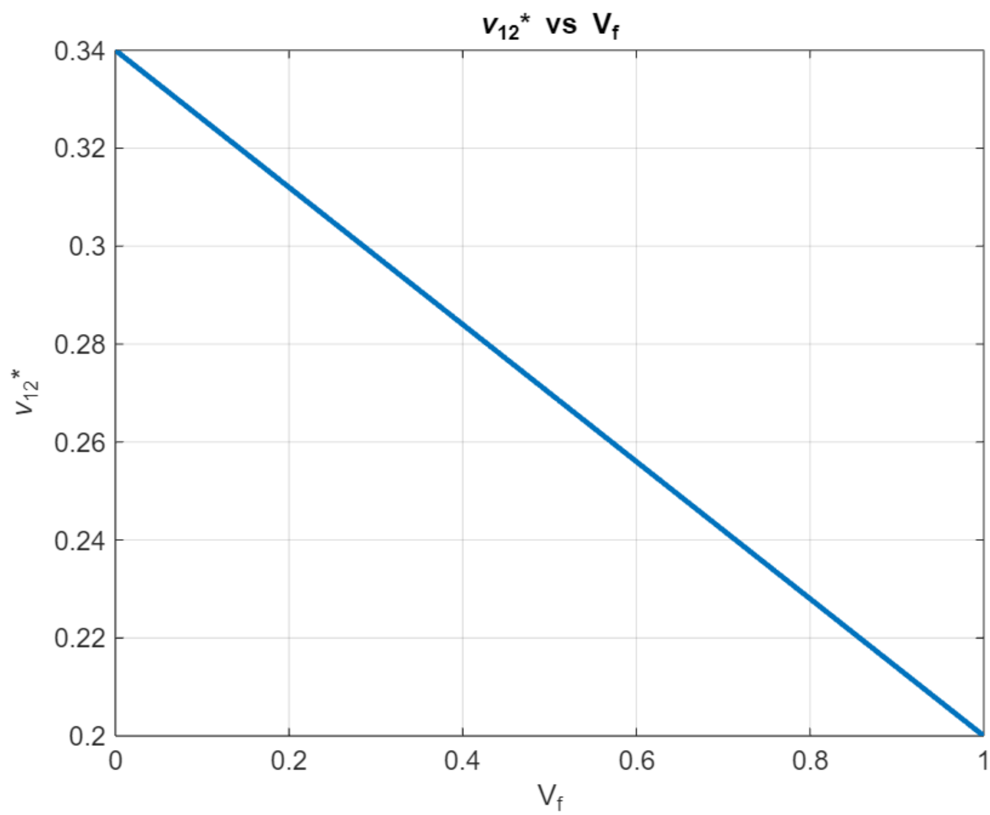
```
%axial youngs modulus
E1_c = E_f*Vf + E_m*(1-Vf);
E1_c_val = subs(E1_c,Vf_val);

%axial poisson's ratio
v12_c = v_f*Vf + v_m*(1-Vf);
v12_c = subs(v12_c,Vf_val);

%direction 2 youngs modulus
E2_c = E_f*E_m / (E_m*Vf + E_f*(1-Vf));
E2_c = subs(E2_c,Vf_val);

%axial shear modulus
```

```
G12_c = G_f*G_m / (G_m*Vf + G_f*(1-Vf));
G12_c = subs(G12_c,Vf_val);

%thermal coeff in direction 1
a1_c = ((a_f*E_f - a_m*E_m)*Vf + a_m*E_m) / ((E_f - E_m)*Vf + E_m);
a1_c = subs(a1_c,Vf_val);

%thermal coeff in direction 2
a2_c = a_f*Vf + a_m*(1-Vf) + ((E_f*v_m - E_m*v_f) / (E1_c)) * (a_m - a_f) * (1-Vf)
* Vf;
a2_c = subs(a2_c,Vf_val);

%plot
% Axial Young's modulus
figure(1)
plot(Vf_val, E1_c_val, 'LineWidth', 2);
grid on;
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
```

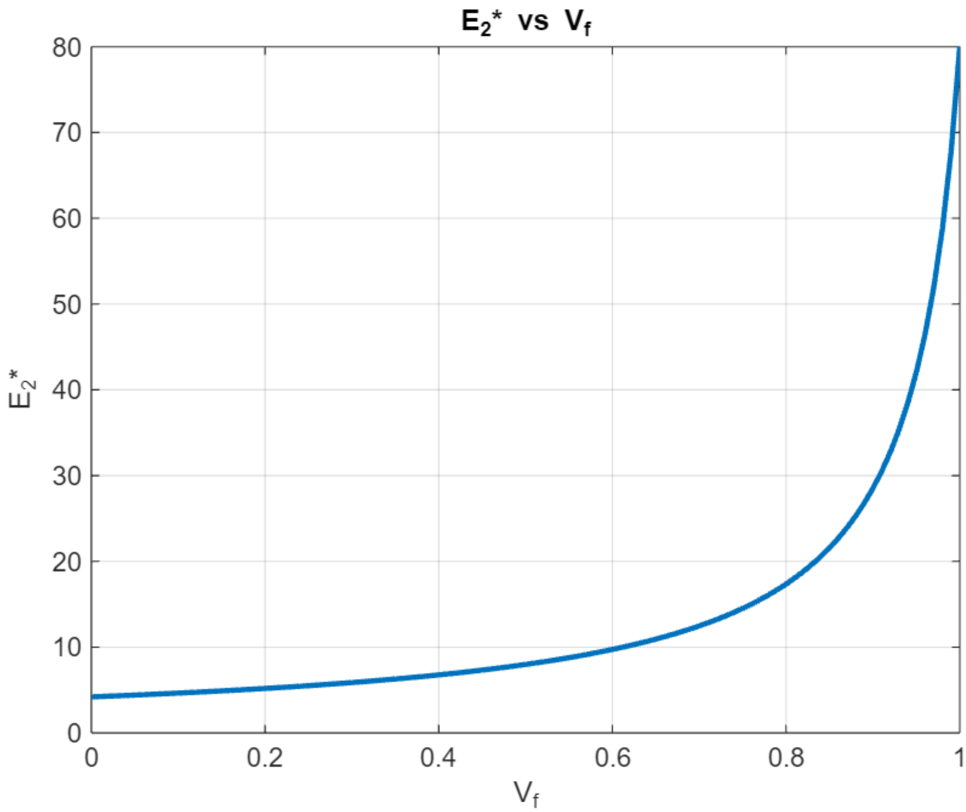



```
% Axial Poisson's ratio
figure(2)
plot(Vf_val, v12_c, 'LineWidth', 2);
grid on;
title('\nu_{12}* vs V_f');
```

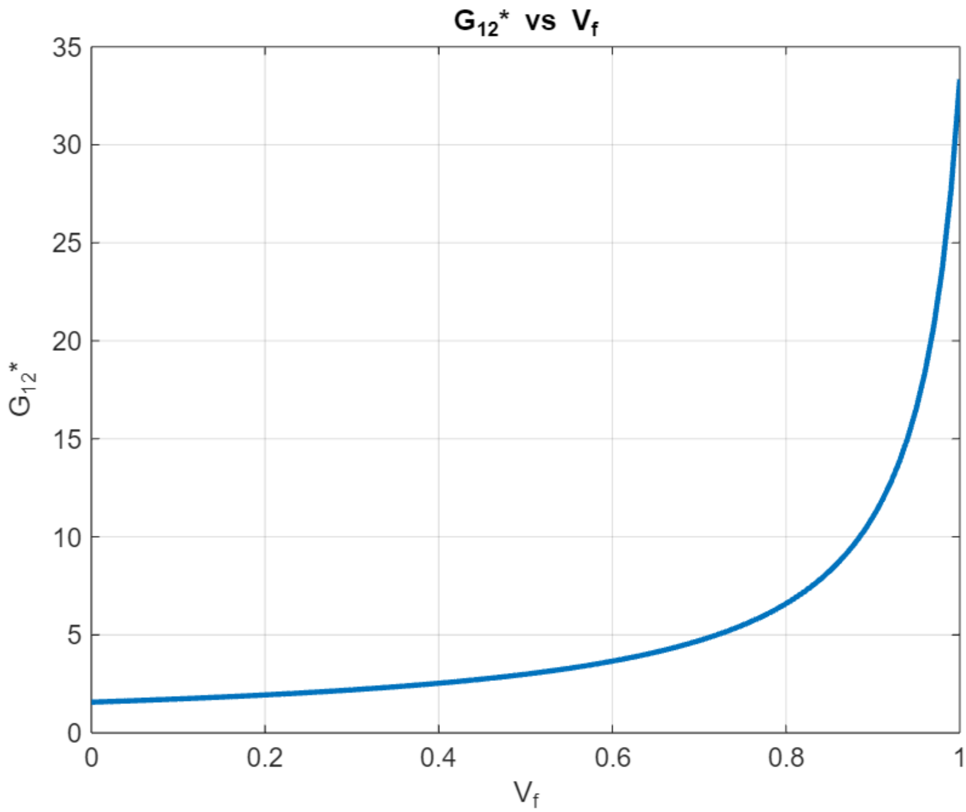
```
xlabel('V_f'); ylabel('\nu_{12}*');
```



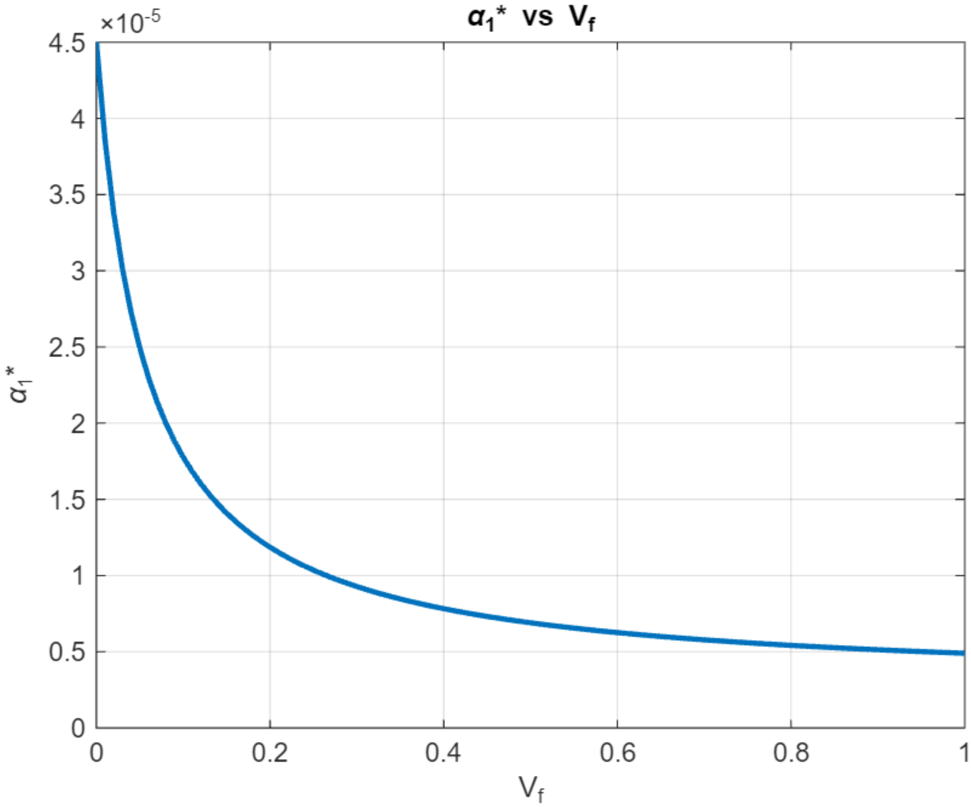

```
% Transverse Young's modulus
figure(3)
plot(Vf_val, E2_c, 'LineWidth', 2);
grid on;
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
```



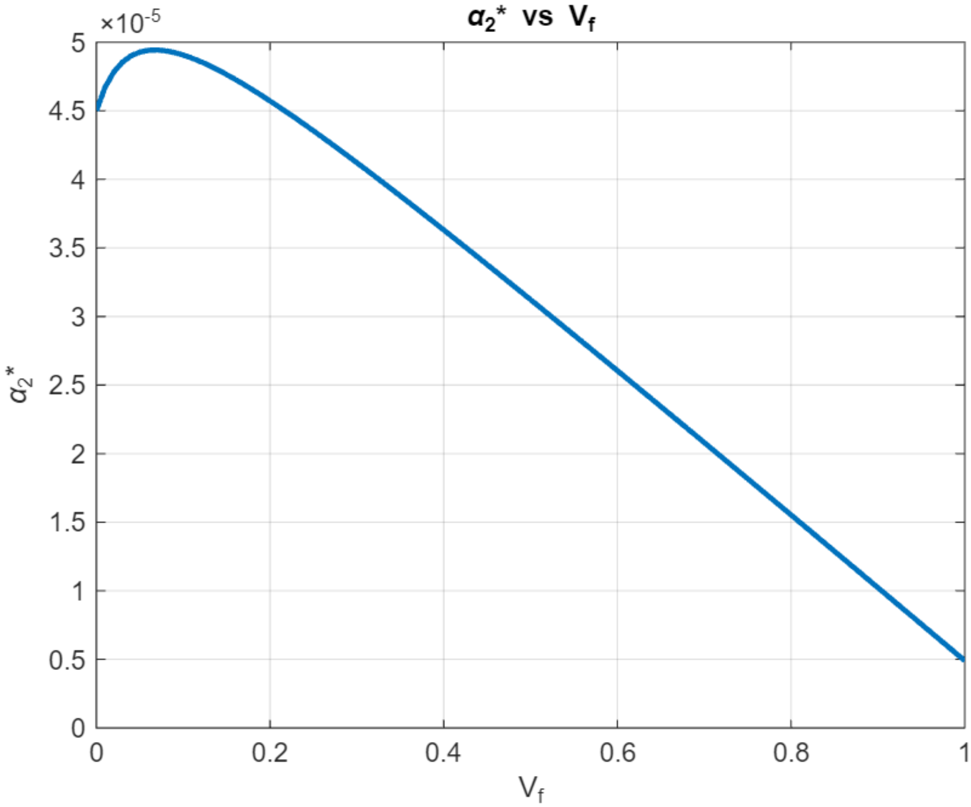

```
% Axial shear modulus
figure(4)
plot(Vf_val, G12_c, 'LineWidth', 2);
grid on;
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
```




```
% Thermal coefficient in direction 1
figure(5)
plot(Vf_val, a1_c, 'LineWidth', 2);
grid on;
xlabel('V_f');
ylabel('\alpha_1*');
title('\alpha_1* vs V_f');
```



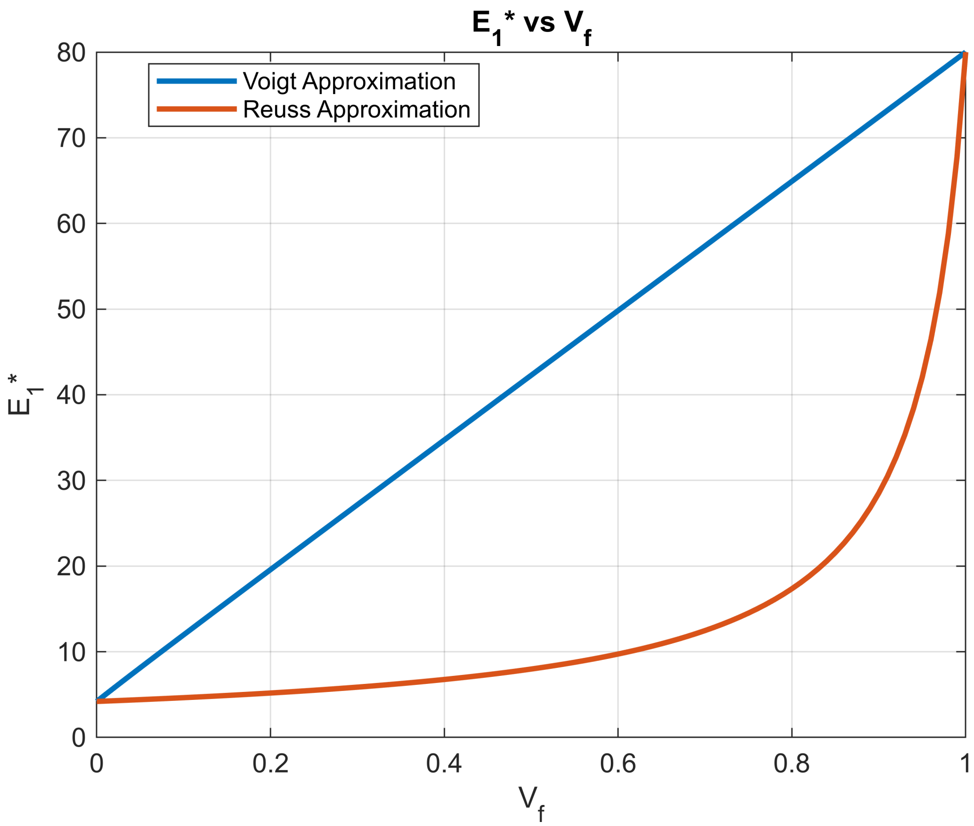

```
% Thermal coefficient in direction 2
figure(6)
plot(Vf_val, a2_c, 'LineWidth', 2);
grid on;
xlabel('V_f');
ylabel('\alpha_2*');
title('\alpha_2* vs V_f');
```


α_2* vs V_f
×10^-5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
α_2*
0
0.2
0.4
0.6
0.8
1
V_f

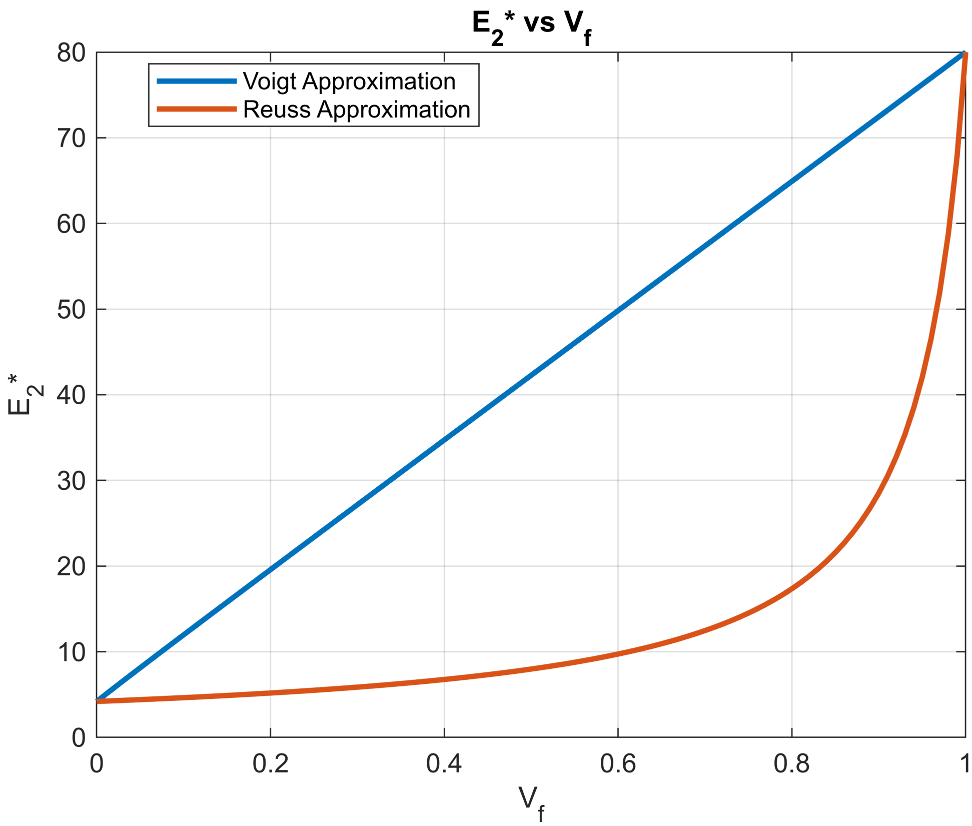
Hills approach

```
clc;clear;
syms Vf
Vf_val = linspace(0,1,101);

E_f = 80;
G_f = 33.33;
v_f = 0.2;
a1_f = 4.9;
a2_f = 4.9;
S_f = [1/E_f, -v_f/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, 1/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, -v_f/E_f, 1/E_f, 0, 0, 0;
       0, 0, 0, 1/G_f, 0, 0;
       0, 0, 0, 0, 1/G_f, 0;
       0, 0, 0, 0, 0, 1/G_f;];
C_f = inv(S_f);

E_m = 4.2;
v_m = 0.34;
a_m = 45;
G_m = E_m/(2*(1+v_m));
S_m = [1/E_m, -v_m/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, 1/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, -v_m/E_m, 1/E_m, 0, 0, 0;
       0, 0, 0, 1/G_m, 0, 0;
       0, 0, 0, 0, 1/G_m, 0;
       0, 0, 0, 0, 0, 1/G_m;];
C_m = inv(S_m);

%voight approximaation
C_c_v = C_m.*(1-Vf) + C_f.*Vf;
S_c_v = inv(C_c_v);

%Reuss approximation
S_c_r = S_m*(1-Vf) + S_f*Vf;

%E1
E1_c_v = 1/S_c_v(1,1);
E1_c_v_val = subs(E1_c_v,Vf_val);
E1_c_r = 1/S_c_r(1,1);
E1_c_r_val = subs(E1_c_r,Vf_val);

%E2
E2_c_v = 1/S_c_v(2,2);
E2_c_v_val = subs(E2_c_v,Vf_val);
E2_c_r = 1/S_c_r(2,2);
E2_c_r_val = subs(E2_c_r,Vf_val);
```

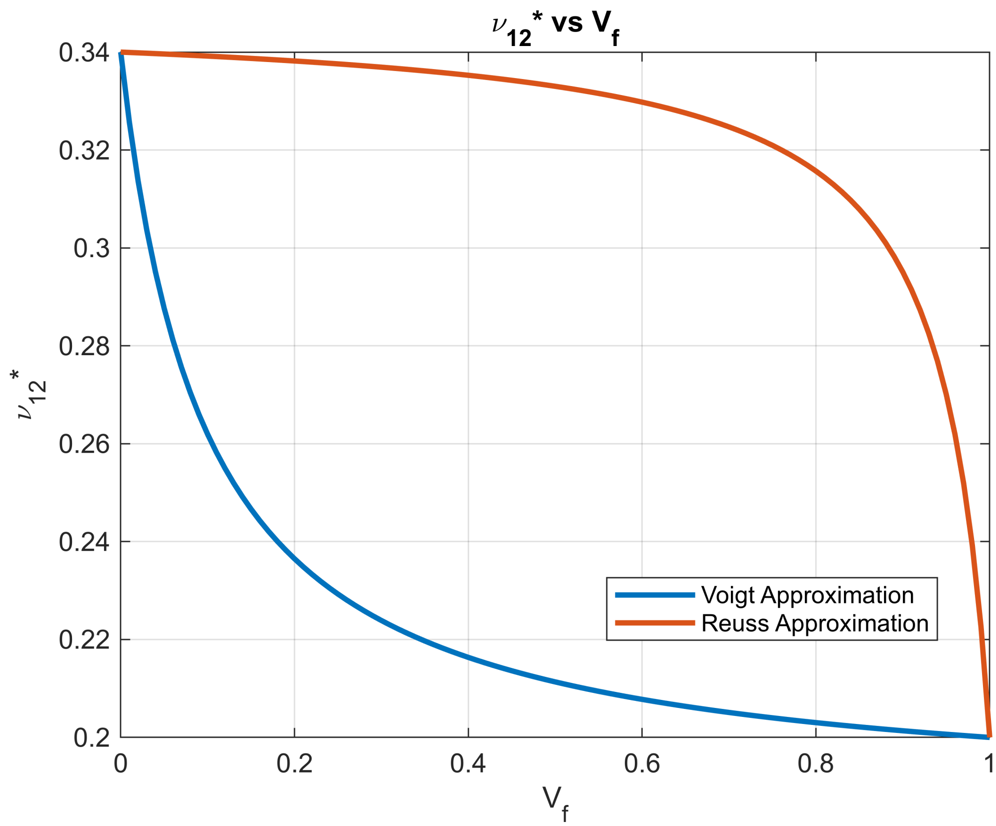
```
%v12
v12_c_v = -S_c_v(1,2)*E1_c_v;
v12_c_v_val = subs(v12_c_v,Vf_val);
v12_c_r = -S_c_r(2,1)*E2_c_r;
v12_c_r_val = subs(v12_c_r,Vf_val);

%v23
v23_c_v = -S_c_v(2,3)*E2_c_v;
v23_c_v_val = subs(v23_c_v,Vf_val);
v23_c_r = -S_c_r(3,2)*E2_c_r;
v23_c_r_val = subs(v23_c_r,Vf_val);

%G12
G12_c_v = 1/S_c_v(5,5);
G12_c_v_val = subs(G12_c_v,Vf_val);
G12_c_r = 1/S_c_r(5,5);
G12_c_r_val = subs(G12_c_r,Vf_val);

%G23
G23_c_v = 1/S_c_v(4,4);
G23_c_v_val = subs(G23_c_v,Vf_val);
G23_c_r = 1/S_c_r(4,4);
G23_c_r_val = subs(G23_c_r,Vf_val);

%plot
%E1
plot(Vf_val,E1_c_v_val,'LineWidth',2);
hold on;
plot(Vf_val,E1_c_r_val,'LineWidth',2);
hold off;
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
legend('Voigt Approximation','Reuss Approximation','Location','best');
grid on;
```



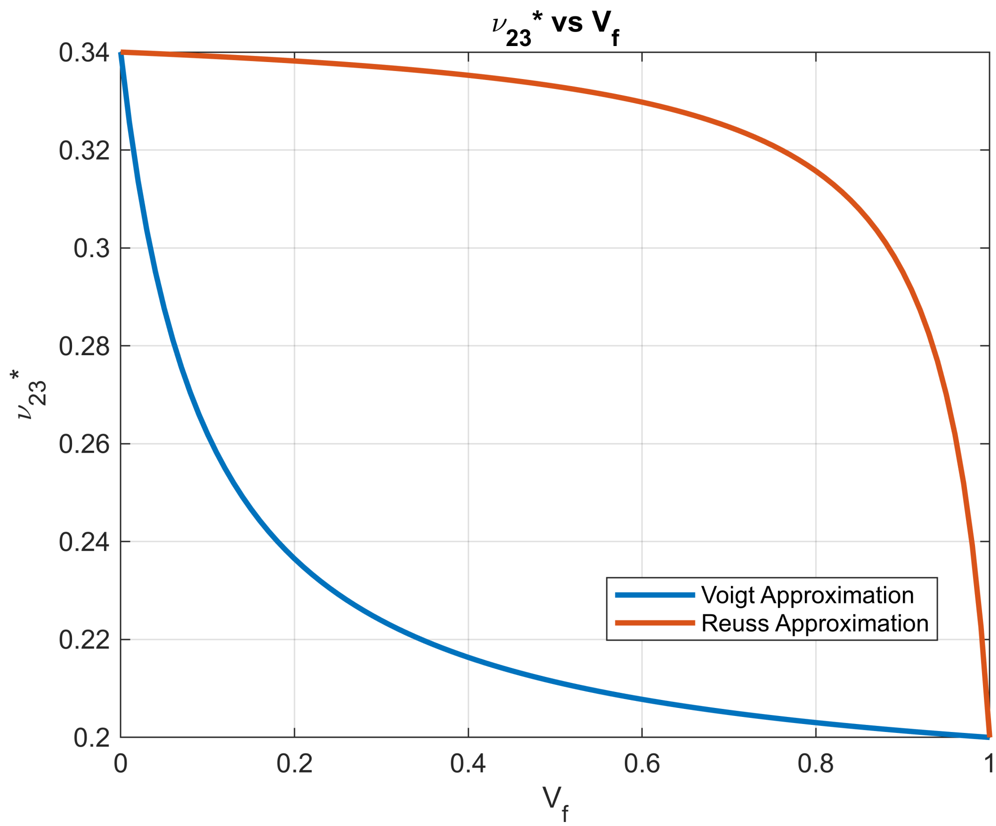

```
%E2
plot(Vf_val,E2_c_v_val,'LineWidth',2);
hold on;
plot(Vf_val,E2_c_r_val,'LineWidth',2);
hold off;
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
legend('Voigt Approximation','Reuss Approximation','Location','best');
grid on;
```



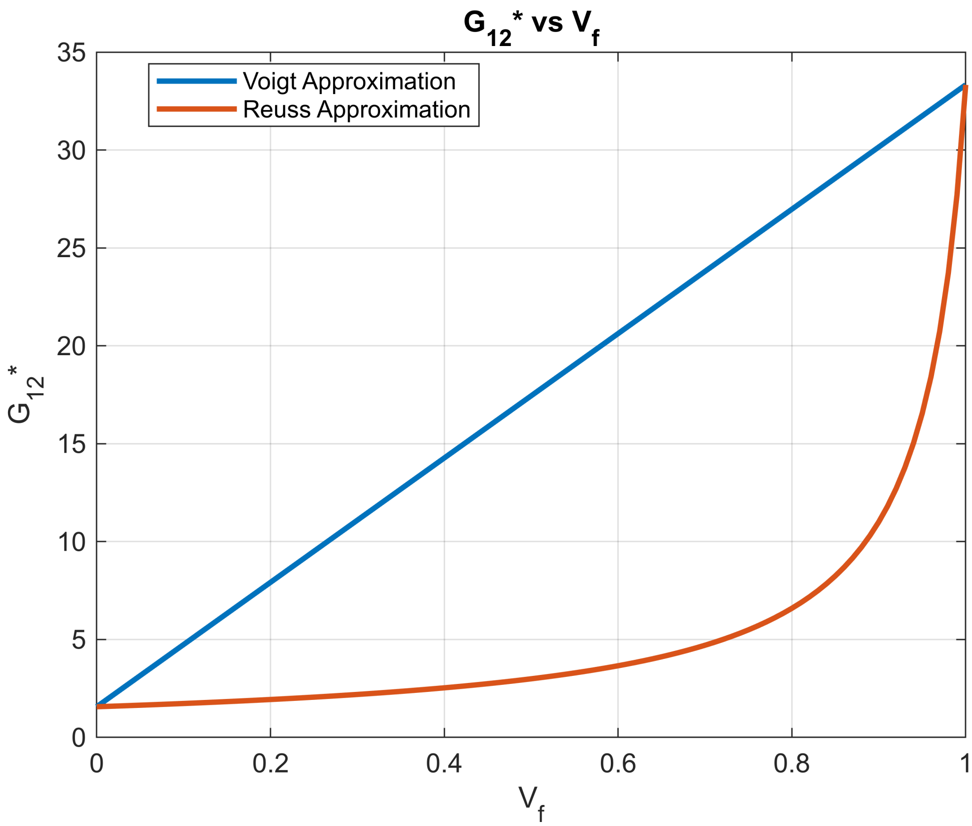

```
%v12
plot(Vf_val,v12_c_v_val,'LineWidth',2);
hold on;
plot(Vf_val,v12_c_r_val,'LineWidth',2);
hold off;
title('\nu_{12}* vs V_f');
xlabel('V_f'); ylabel('\nu_{12}*');
legend('Voigt Approximation','Reuss Approximation','Location','best');
grid on;
```



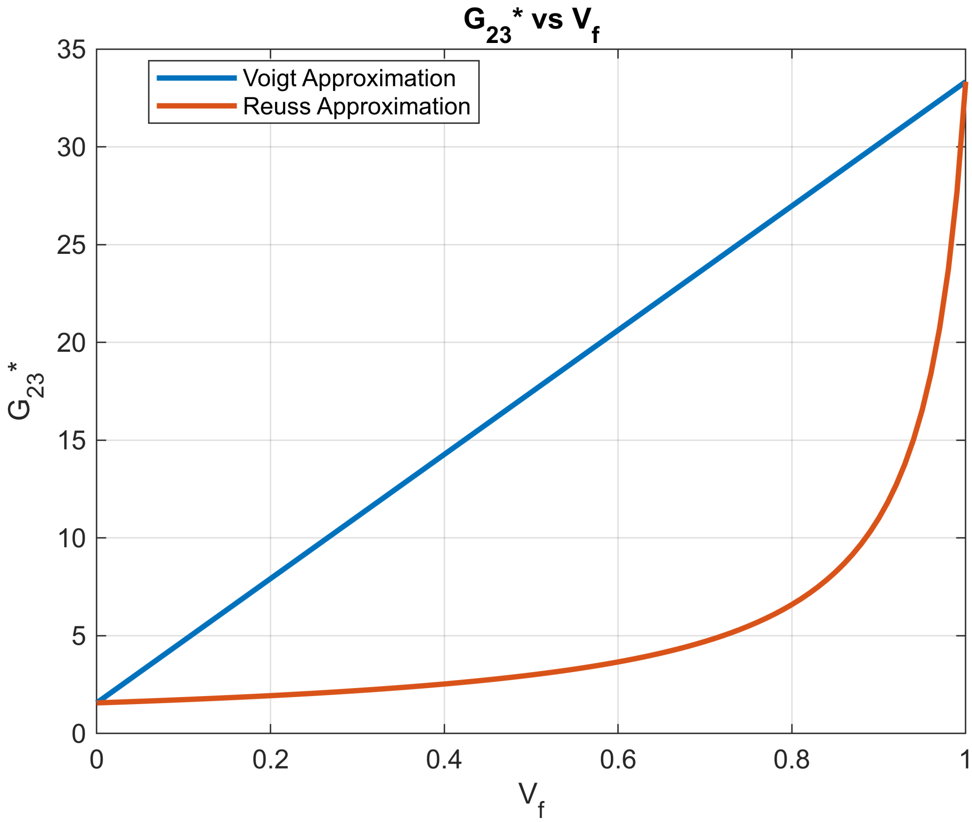

```
%v23
plot(Vf_val,v23_c_v_val,'LineWidth',2);
hold on;
plot(Vf_val,v23_c_r_val,'LineWidth',2);
hold off;
title('\nu_{23}* vs V_f');
xlabel('V_f'); ylabel('\nu_{23}*');
legend('Voigt Approximation','Reuss Approximation','Location','best');
grid on;
```




```
%G12
plot(Vf_val,G12_c_v_val,'LineWidth',2);
hold on;
plot(Vf_val,G12_c_r_val,'LineWidth',2);
hold off;
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
legend('Voigt Approximation','Reuss Approximation','Location','best');
grid on;
```




```
%G23
plot(Vf_val,G23_c_v_val,'LineWidth',2);
hold on;
plot(Vf_val,G23_c_r_val,'LineWidth',2);
hold off;
title('G_{23}* vs V_f');
xlabel('V_f'); ylabel('G_{23}*');
legend('Voigt Approximation','Reuss Approximation','Location','best');
grid on;
```

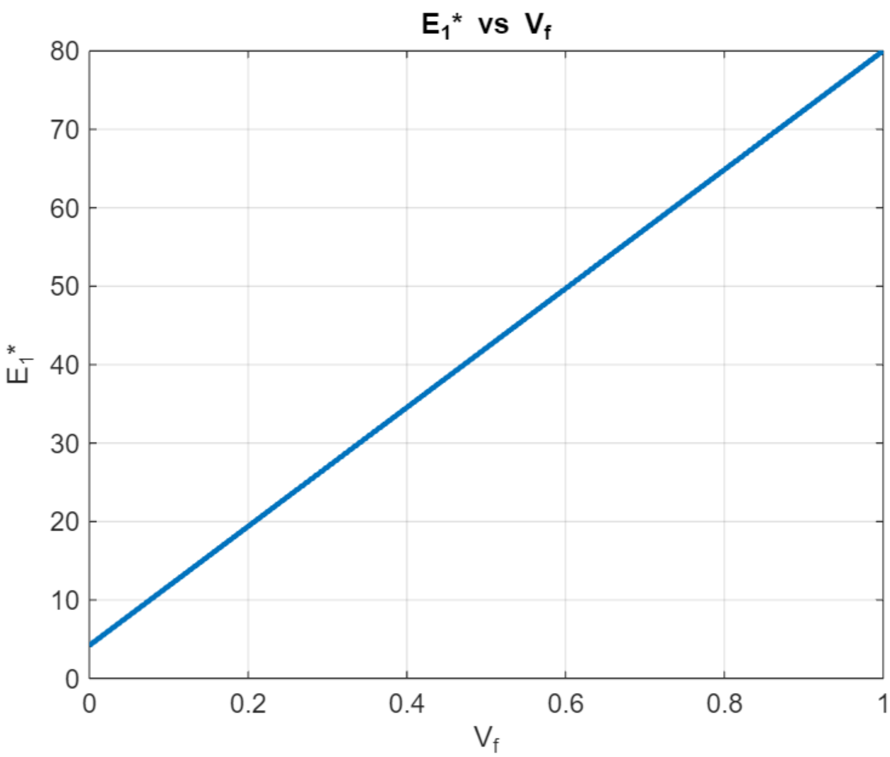


G_{23}* vs V_f
Voigt Approximation
Reuss Approximation
35
30
25
20
15
10
5
0
G_{23}*
0
0.2
0.4
0.6
0.8
1
V_f

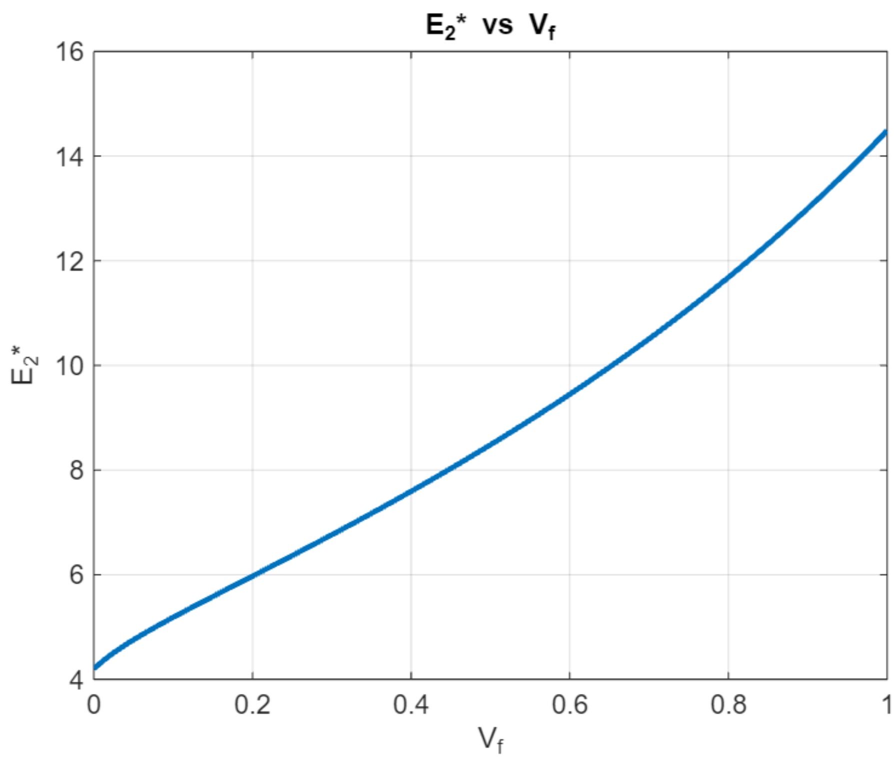
Concentric Cylinder Assemblage Model

```
clc;clear;
syms Vf
Vf_val = linspace(0,1,101);

% Fiber properties
E_f = 80;
G_f = 33.33;
v_f = 0.2;
k_f = E_f/(3*(1-2*v_f));

% Matrix properties
E_m = 4.2;
v_m = 0.34;
G_m = E_m/(2*(1+v_m));
k_m = E_m/(3*(1-2*v_m));

%E1
A = (1 - Vf)*G_m/(k_f + 0.5*G_f);
B = Vf*G_m/(k_m + G_m/3);
E1_c = Vf*E_f + (1 - Vf)*E_m + (4*Vf*(1-Vf)*(v_f - v_m)^2*G_m)/(A + B + 1);
E1_c_val = subs(E1_c,Vf_val);

%v12
v12_c = (1-Vf)*v_m + Vf*v_f + (Vf*(1-Vf)*(v_f-v_m)*(G_m/(k_m+G_m/3) - G_m/(k_f+G_f/
3)))/(A + B +1);
v12_c_val = subs(v12_c,Vf_val);

%k23
k23_c = k_m + G_m/3 + Vf/(1/((k_f-k_m) + (G_f - G_m)/3) + (1-Vf)/(k_m+4*G_m/3));
k23_c_val = subs(k23_c,Vf_val);

%G12
G12_c = G_m*((G_f*(1+Vf)+G_m*(1-Vf))/(G_f*(1-Vf)+G_m*(1+Vf)));
G12_c_val = subs(G12_c,Vf_val);

%G23
A = (G_m/(G_f - G_m) + (k_m + 7*G_m/3)/(2*k_m + 8*G_m/3));
G23_c = G_m*(1 + Vf/A);
G23_c_val = subs(G23_c,Vf_val);

% Formula for E2_c
E2_c = (4*G23_c*k23_c)/(k23_c + G23_c + (4*(v12_c)^2*G23_c*k23_c)/E1_c);
E2_c_val = subs(E2_c,Vf_val);

% Formula for v23_c
v23_c = (k23_c - G23_c - (4*(v12_c)^2*G23_c*k23_c)/E1_c)/(k23_c + G23_c +
4*(v12_c)^2*G23_c*k23_c/E1_c);
v23_c_val = subs(v23_c,Vf_val);
```

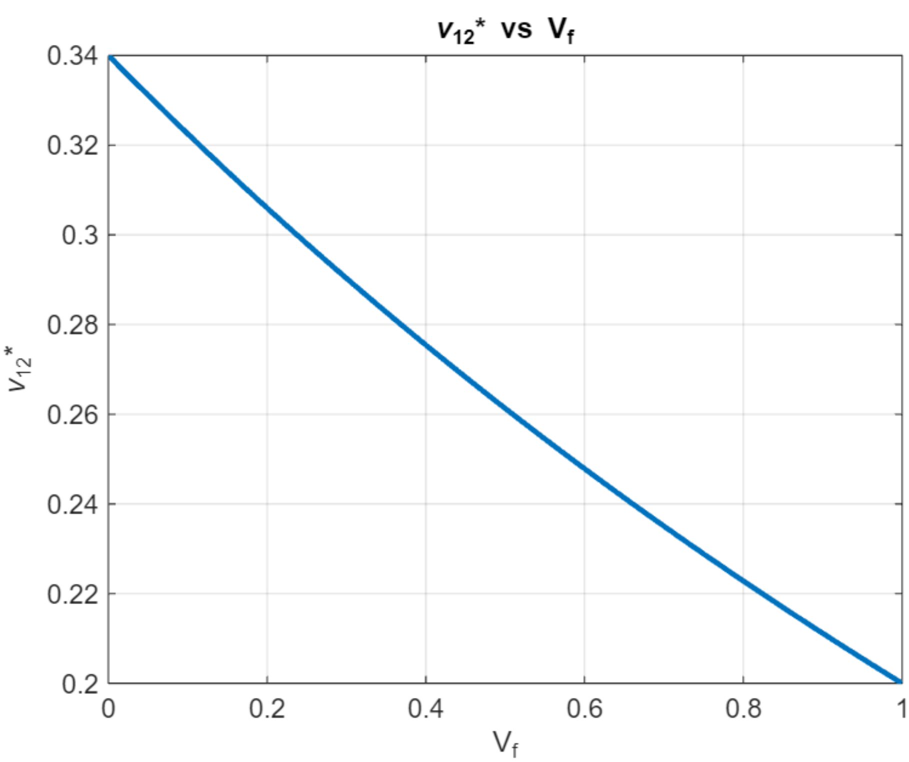
```
% Formula for v21_c
v21_c = (4*v12_c*G23_c*k23_c)/(E1_c*(k23_c + G23_c) + 4*(v12_c)^2*G23_c*k23_c);
v21_c_val = subs(v21_c,Vf_val);

% Plotting
figure;
plot(Vf_val, E1_c_val, 'LineWidth', 2);
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
grid on;
```



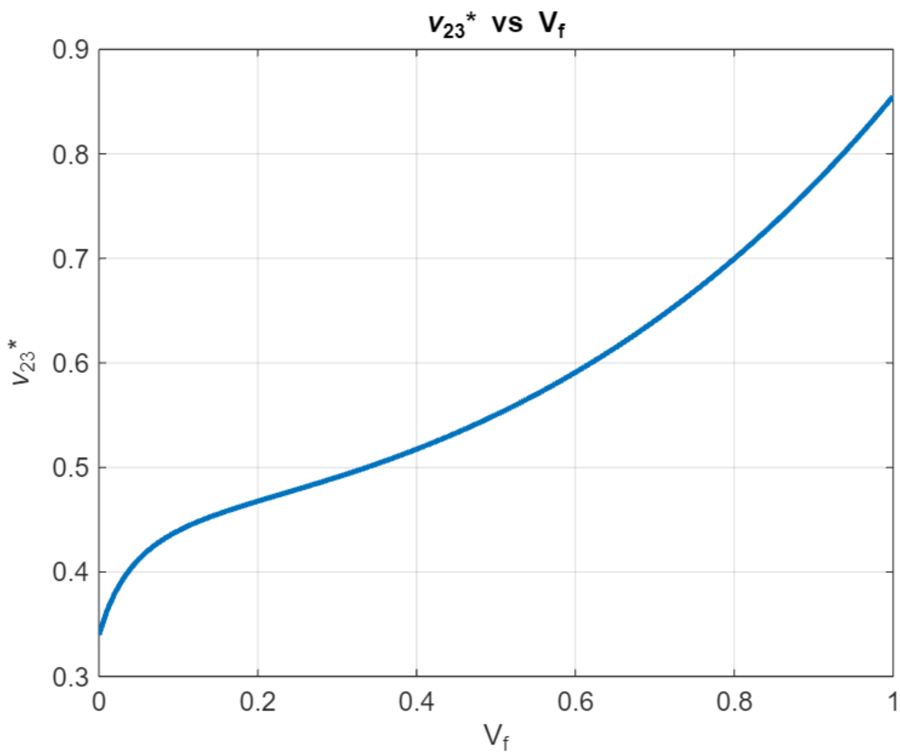

```
figure;
plot(Vf_val, E2_c_val, 'LineWidth', 2);
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
grid on;
```



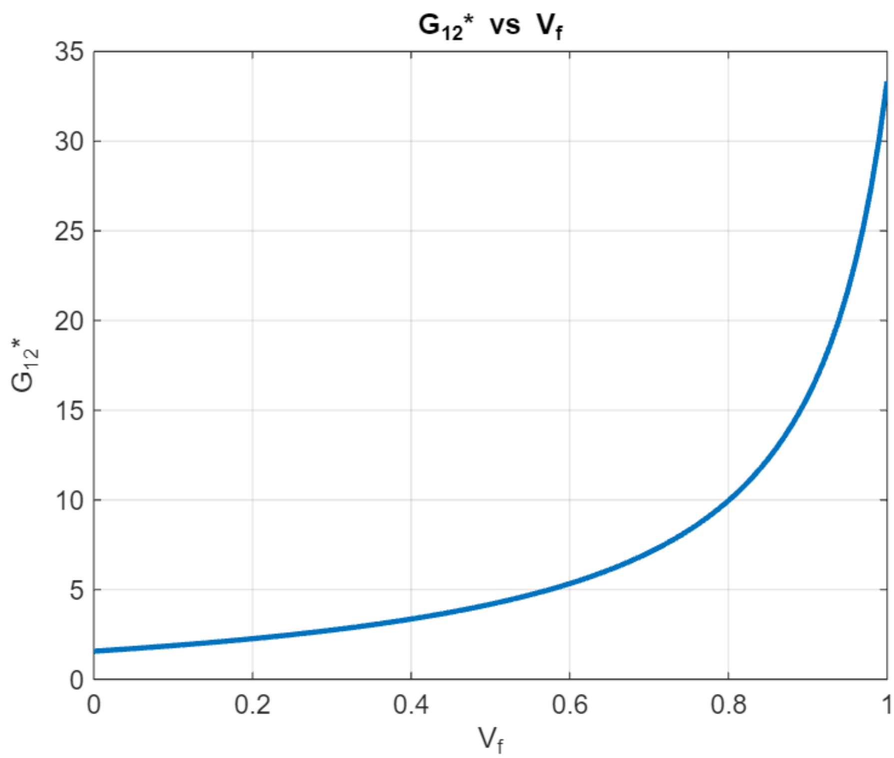

```
figure;
plot(Vf_val, v12_c_val, 'LineWidth', 2);
title('\nu_{12}* vs V_f');
xlabel('V_f'); ylabel('\nu_{12}*');
grid on;
```



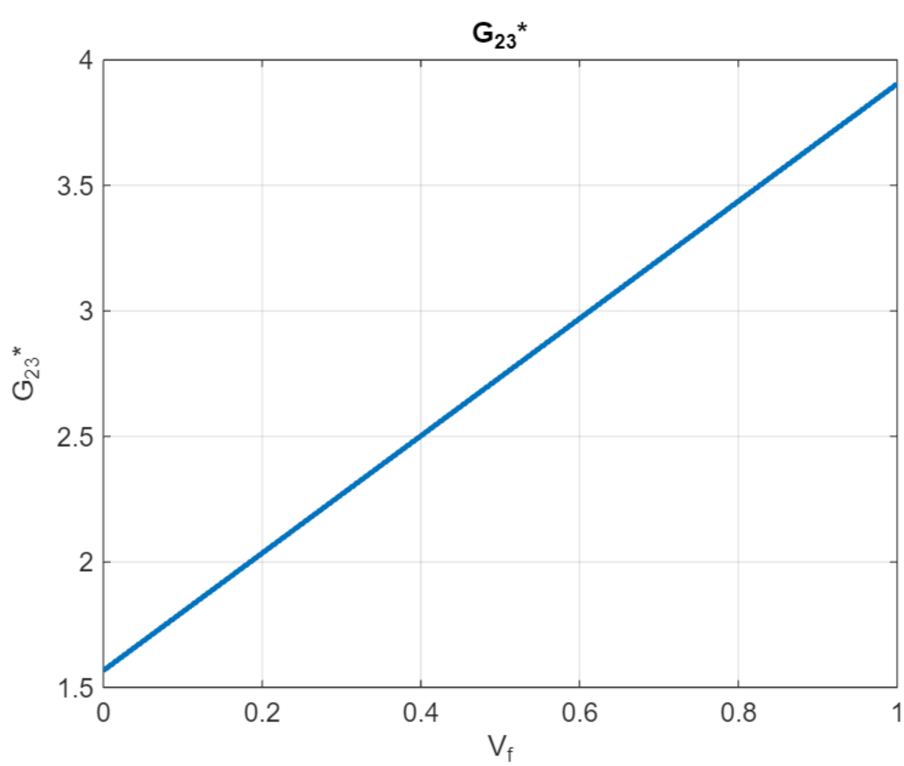
```
figure;
plot(Vf_val, v23_c_val, 'LineWidth', 2);
title('\nu_{23}* vs V_f');
xlabel('V_f'); ylabel('\nu_{23}*');
grid on;
```



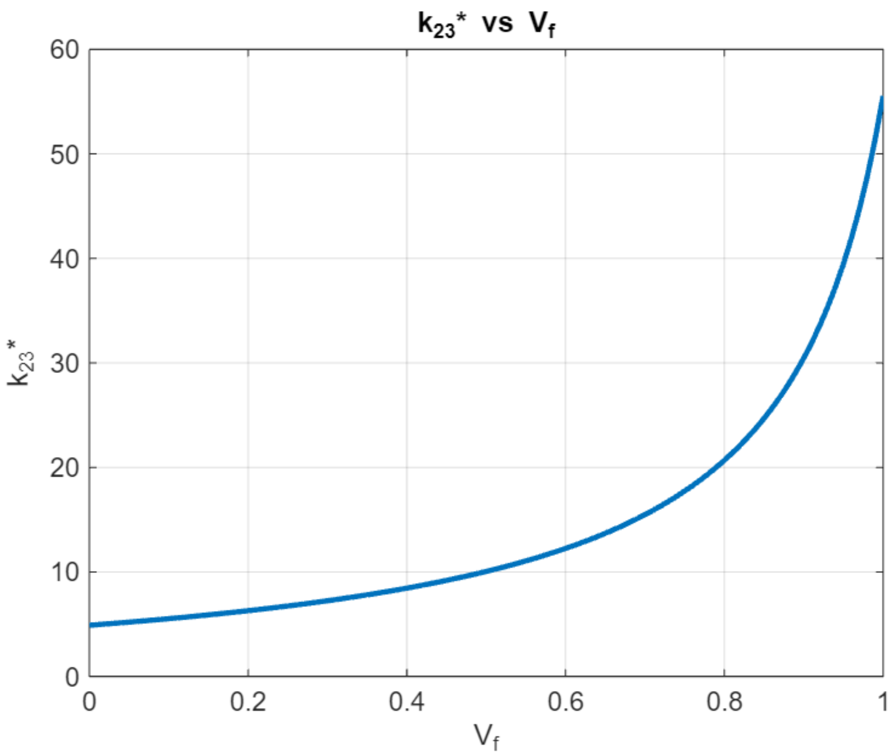

```
figure;
plot(Vf_val, G12_c_val, 'LineWidth', 2);
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
grid on;
```



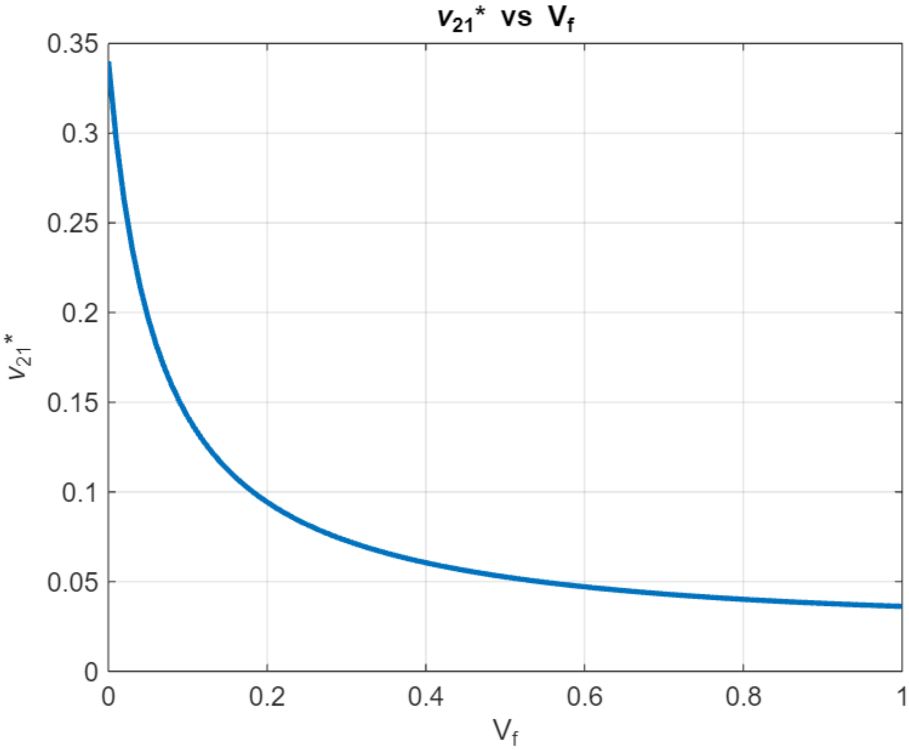

```
figure;
plot(Vf_val, G23_c_val, 'LineWidth', 2);
title('G_{23}* vs V_f');
xlabel('V_f'); ylabel('G_{23}*');
grid on;
```



```
figure;
plot(Vf_val, k23_c_val, 'LineWidth', 2);
title('k_{23}* vs V_f');
xlabel('V_f'); ylabel('k_{23}*');
grid on;
```




```
figure;
plot(Vf_val, v21_c_val, 'LineWidth', 2);
title('\nu_{21}* vs V_f');
xlabel('V_f'); ylabel('\nu_{21}*');
grid on;
```


v21* vs Vf
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
v21*
0
0.2
0.4
0.6
0.8
1
Vf

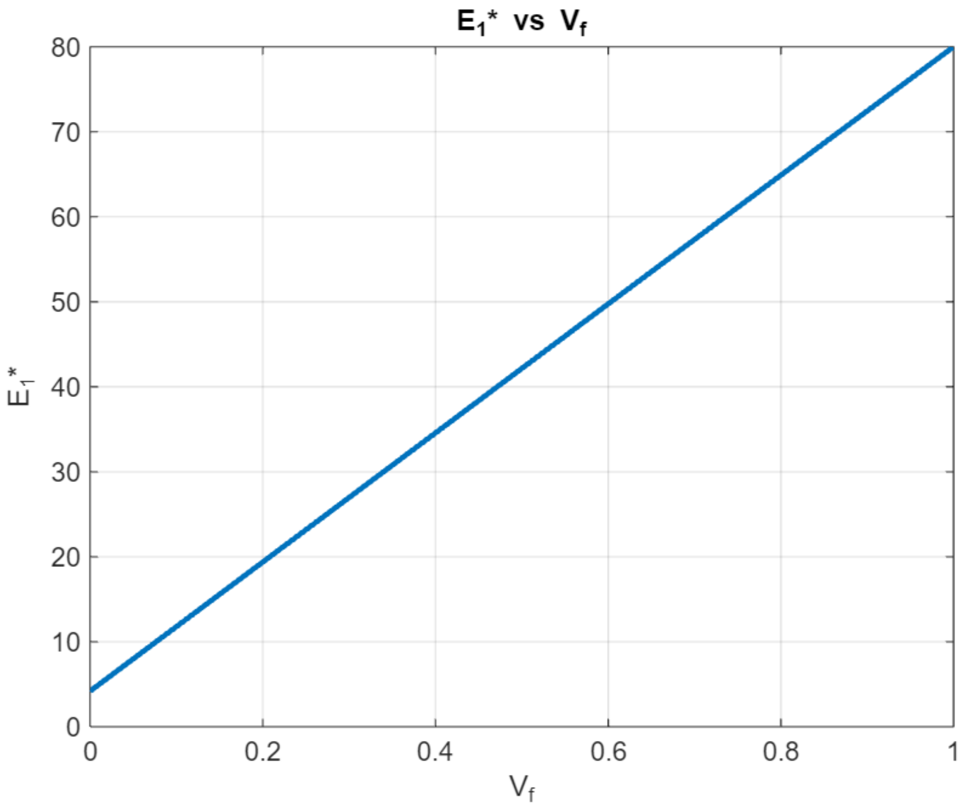
Self-consistent Method

```
clc;clear;
Vf_val = linspace(0, 1, 101);
Vm_val = 1 - Vf_val;

E_f = 80;
G_f = 33.33;
v_f = 0.2;

S_f = [1/E_f, -v_f/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, 1/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, -v_f/E_f, 1/E_f, 0, 0, 0;
       0, 0, 0, 1/G_f, 0, 0;
       0, 0, 0, 0, 1/G_f, 0;
       0, 0, 0, 0, 0, 1/G_f;];
C_f = inv(S_f);

E_m = 4.2;
v_m = 0.34;
G_m = E_m/(2*(1+v_m));

S_m = [1/E_m, -v_m/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, 1/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, -v_m/E_m, 1/E_m, 0, 0, 0;
       0, 0, 0, 1/G_m, 0, 0;
       0, 0, 0, 0, 1/G_m, 0;
       0, 0, 0, 0, 0, 1/G_m;];
C_m = inv(S_m);

% fiber
syms n_f m_f k_f l_f p_f
eqn1 = n_f == C_f(1,1);
eqn2 = l_f == C_f(1,2);
eqn3 = k_f + m_f == C_f(2,2);
eqn4 = m_f == C_f(4,4);
eqn5 = p_f == C_f(5,5);

fiber = solve([eqn1,eqn2,eqn3,eqn4,eqn5],[n_f,l_f,k_f,m_f,p_f]);
n_f = double(fiber.n_f);
m_f = double(fiber.m_f);
k_f = double(fiber.k_f);
l_f = double(fiber.l_f);
p_f = double(fiber.p_f);

% matrix
syms n_m m_m k_m l_m p_m
eqn1 = n_m == C_m(1,1);
eqn2 = l_m == C_m(1,2);
eqn3 = k_m + m_m == C_m(2,2);
```

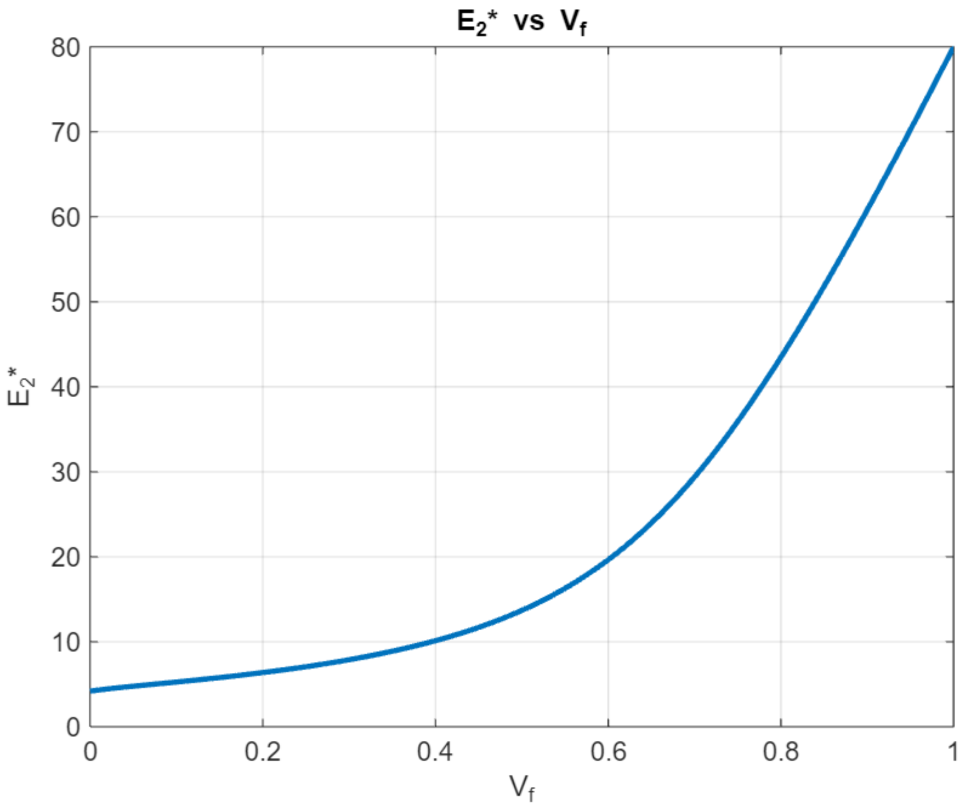
```
eqn4 = m_m == C_m(4,4);
eqn5 = p_m == C_m(5,5);

matrix = solve([eqn1,eqn2,eqn3,eqn4,eqn5],[n_m,l_m,k_m,m_m,p_m]);
n_m = double(matrix.n_m);
m_m = double(matrix.m_m);
k_m = double(matrix.k_m);
l_m = double(matrix.l_m);
p_m = double(matrix.p_m);

% Storage for results
k_c = zeros(size(Vf_val));
m_c = zeros(size(Vf_val));
p_c = zeros(size(Vf_val));


k_c(1) = k_m;
m_c(1) = l_m;
p_c(1) = p_m;

k_c(length(Vf_val)) = k_f;
m_c(length(Vf_val)) = l_f;
p_c(length(Vf_val)) = p_f;

for i = 2:length(Vf_val)-1
    current_Vf = Vf_val(i);
    current_Vm = Vm_val(i);

    syms k_star m_star p_star

    eq1 = (current_Vf * k_f) / (k_f + m_star) + (current_Vm * k_m) / (k_m + m_star)
== ...
          2 * ((current_Vf * m_m) / (m_m - m_star) + (current_Vm * m_f) / (m_f -
m_star));

    eq2 = (1 / (2 * p_star)) == (current_Vf / (2 * (p_star - p_m))) + (current_Vm /
(2 * (p_star - p_f)));

    eq3 = (1 / (k_star + m_star)) == (current_Vf / (k_f + m_star)) + (current_Vm /
(k_m + m_star));

    sol = solve([eq1, eq2, eq3], [k_star, m_star, p_star]);

    k_star_sol = double(sol.k_star);
    m_star_sol = double(sol.m_star);
    p_star_sol = double(sol.p_star);

    approved = find(isreal(k_star_sol) & isreal(m_star_sol) & isreal(p_star_sol) &
...
                     (k_star_sol >= 0) & (m_star_sol >= 0) & (p_star_sol >= 0));
```

```
    k_c(i) = k_star_sol(approved(1));
    m_c(i) = m_star_sol(approved(1));
    p_c(i) = p_star_sol(approved(1));
end

l_c = zeros(size(Vf_val));
n_c = zeros(size(Vf_val));

l_c(1) = l_m;
l_c(end) = l_f;

n_c(1) = n_m;
n_c(end) = n_f;

for i = 2:length(Vf_val)-1
    current_Vf = Vf_val(i);
    current_Vm = Vm_val(i);

    k_star = k_c(i);
    m_star = m_c(i);

    A = (k_f - k_m) / (l_f - l_m);

    l_star = l_f + (k_star - k_f)/A;

    n_star = (l_star - current_Vf * l_f - current_Vm * l_m)/A + ...
             current_Vf * n_f + current_Vm * n_m;

    l_c(i) = l_star;
    n_c(i) = n_star;
end

C_c_all = cell(size(Vf_val));
S_c_all = cell(size(Vf_val));

for i = 1:length(Vf_val)

    C_c = [n_c(i), l_c(i), l_c(i), 0,      0,      0;
           l_c(i), k_c(i)+m_c(i), k_c(i)-m_c(i), 0, 0, 0;
           l_c(i), k_c(i)-m_c(i), k_c(i)+m_c(i), 0, 0, 0;
           0,      0,      0,      m_c(i), 0,      0;
           0,      0,      0,      0,      p_c(i), 0;
           0,      0,      0,      0,      0,      p_c(i)];

    S_c = inv(C_c);

    C_c_all{i} = C_c;
    S_c_all{i} = S_c;
end
```

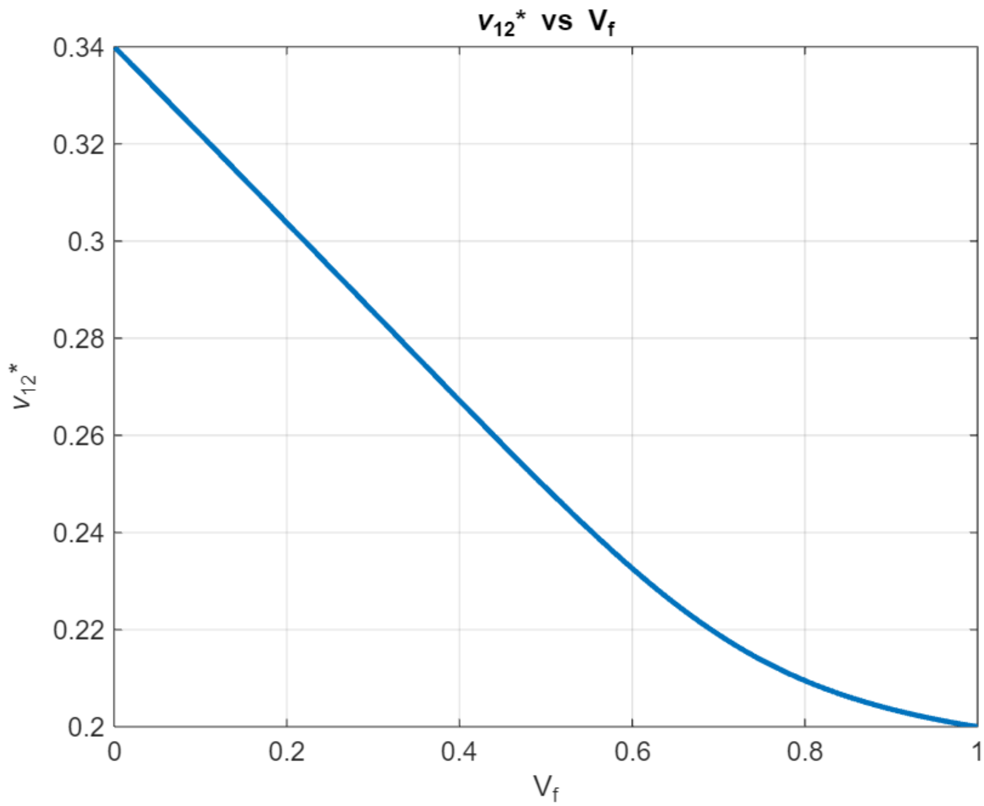
```matlab
E1_c_val   = zeros(size(Vf_val));
E2_c_val   = zeros(size(Vf_val));
v12_c_val  = zeros(size(Vf_val));
v23_c_val  = zeros(size(Vf_val));
G12_c_val  = zeros(size(Vf_val));
G23_c_val  = zeros(size(Vf_val));

E1_c_val(1)  = E_m;
E2_c_val(1)  = E_m;
v12_c_val(1) = v_m;
v23_c_val(1) = v_m;
G12_c_val(1) = G_m;
G23_c_val(1) = G_m;

E1_c_val(length(Vf_val))  = E_f;
E2_c_val(length(Vf_val))  = E_f;
v12_c_val(length(Vf_val)) = v_f;
v23_c_val(length(Vf_val)) = v_f;
G12_c_val(length(Vf_val)) = G_f;
G23_c_val(length(Vf_val)) = G_f;

for i = 2:length(Vf_val)-1
    S = S_c_all{i};

    E1_c_val(i)  = 1 / S(1,1);
    E2_c_val(i)  = 1 / S(2,2);
    v12_c_val(i) = -S(1,2) * E1_c_val(i);
    v23_c_val(i) = -S(2,3) * E2_c_val(i);
    G12_c_val(i) = 1 / S(5,5);
    G23_c_val(i) = 1 / S(4,4);
end

% Plotting
figure;
plot(Vf_val, E1_c_val, 'LineWidth', 2);
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
grid on;
```



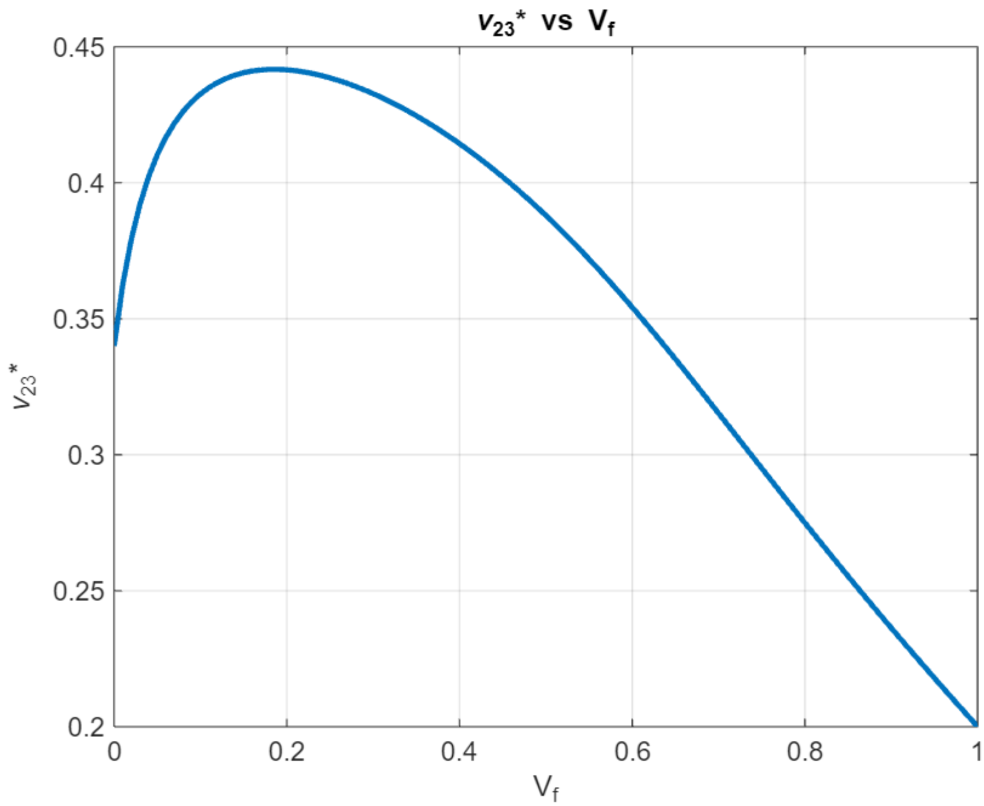

```
figure;
plot(Vf_val, E2_c_val, 'LineWidth', 2);
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
grid on;
```



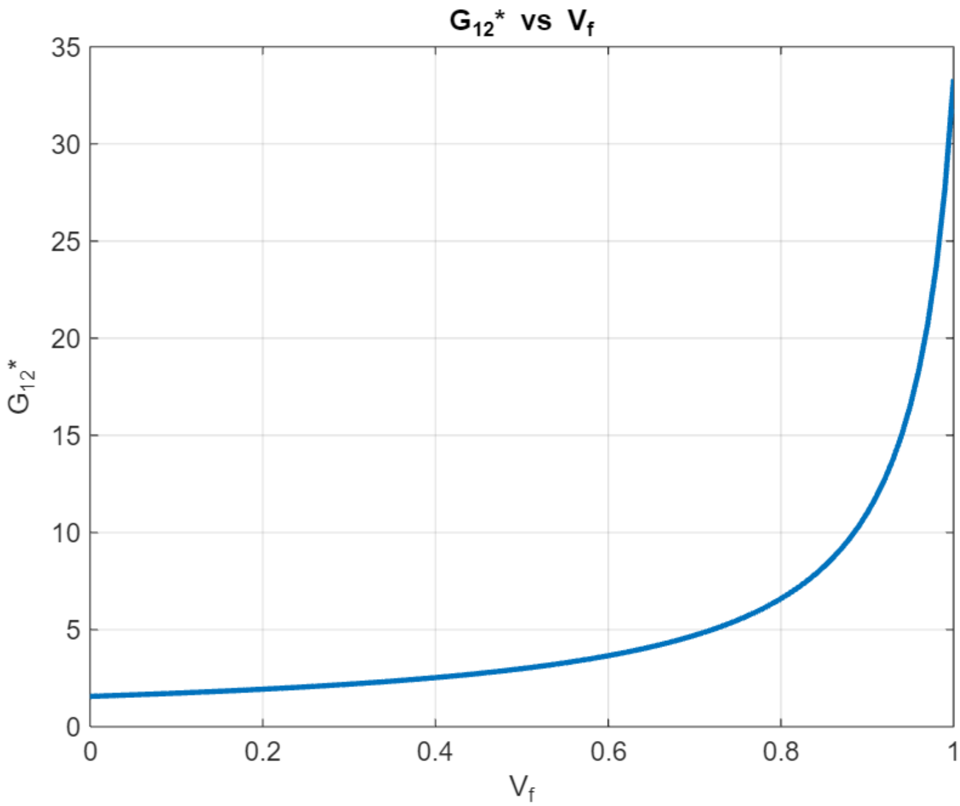

```
figure;
plot(Vf_val, v12_c_val, 'LineWidth', 2);
title('\nu_{12}* vs V_f');
xlabel('V_f'); ylabel('\nu_{12}*');
grid on;
```



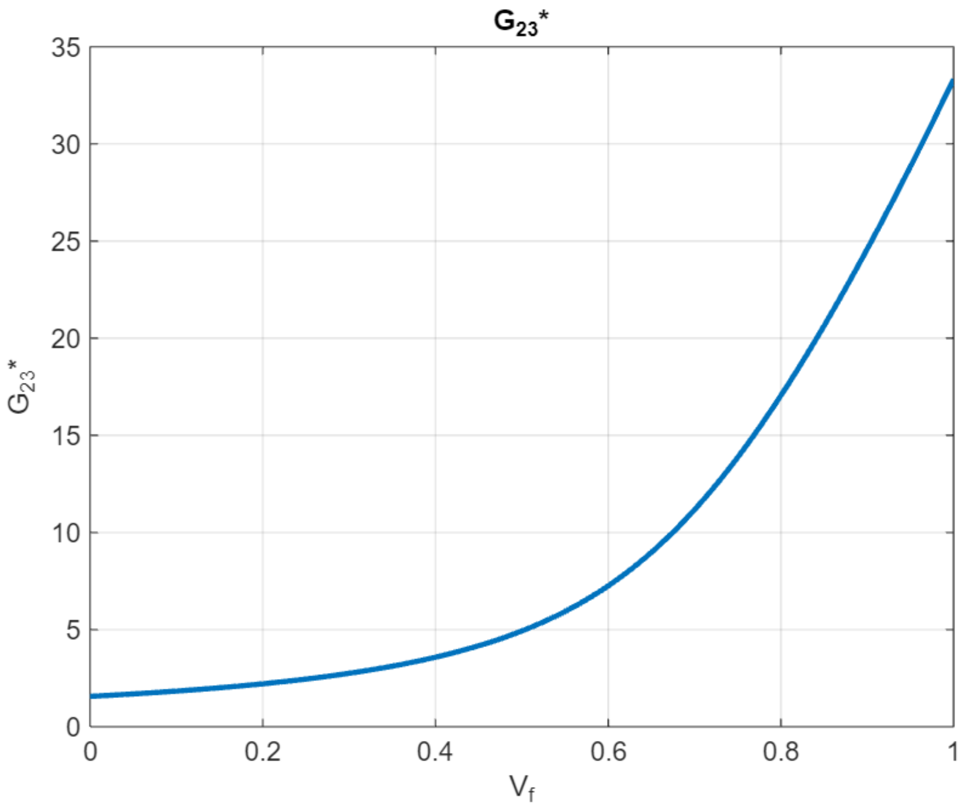

```
figure;
plot(Vf_val, v23_c_val, 'LineWidth', 2);
title('\nu_{23}* vs V_f');
xlabel('V_f'); ylabel('\nu_{23}*');
grid on;
```




```
figure;
plot(Vf_val, G12_c_val, 'LineWidth', 2);
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
grid on;
```




```
figure;
plot(Vf_val, G23_c_val, 'LineWidth', 2);
title('G_{23}* vs V_f');
xlabel('V_f'); ylabel('G_{23}*');
grid on;
```



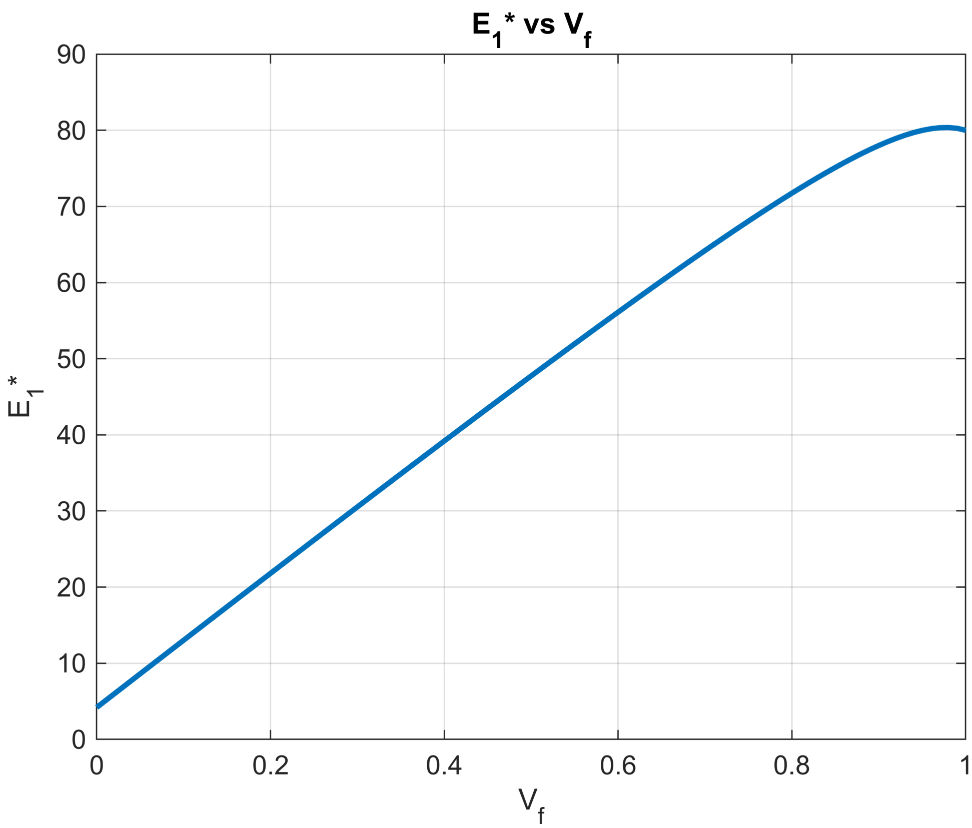
G23*
35
30
25
20
15
10
5
0
G23*
0
0.2
0.4
0.6
0.8
1
Vf

Mori-Tanaka Method

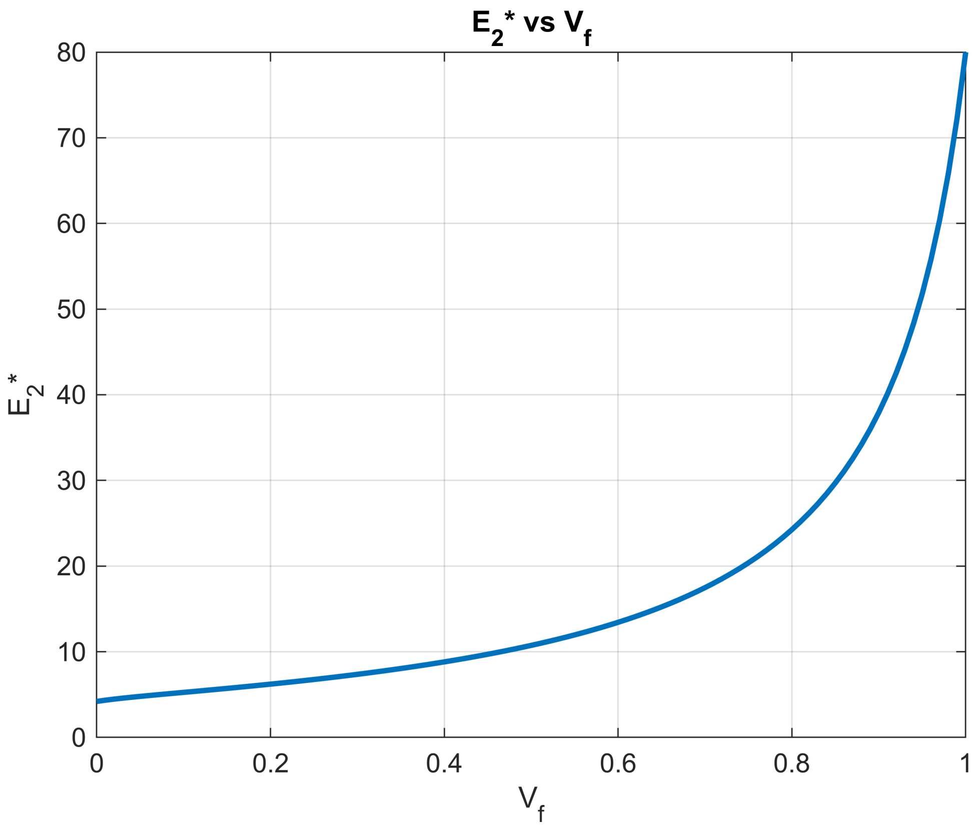
```
clc;clear;
syms Vf
Vf_val = linspace(0,1,101);

E_f = 80;
G_f = 33.33;
v_f = 0.2;

S_f = [1/E_f, -v_f/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, 1/E_f, -v_f/E_f, 0, 0, 0;
       -v_f/E_f, -v_f/E_f, 1/E_f, 0, 0, 0;
       0, 0, 0, 1/G_f, 0, 0;
       0, 0, 0, 0, 1/G_f, 0;
       0, 0, 0, 0, 0, 1/G_f;];
C_f = inv(S_f);

E_m = 4.2;
v_m = 0.34;
G_m = E_m/(2*(1+v_m));

S_m = [1/E_m, -v_m/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, 1/E_m, -v_m/E_m, 0, 0, 0;
       -v_m/E_m, -v_m/E_m, 1/E_m, 0, 0, 0;
       0, 0, 0, 1/G_m, 0, 0;
       0, 0, 0, 0, 1/G_m, 0;
       0, 0, 0, 0, 0, 1/G_m;];
C_m = inv(S_m);

% fiber
syms n_f m_f k_f l_f p_f
eqn1 = n_f == C_f(1,1);
eqn2 = l_f == C_f(1,2);
eqn3 = k_f + m_f == C_f(2,2);
eqn4 = m_f == C_f(4,4);
eqn5 = p_f == C_f(5,5);

fiber = solve([eqn1,eqn2,eqn3,eqn4,eqn5],[n_f,l_f,k_f,m_f,p_f]);
n_f = double(fiber.n_f);
m_f = double(fiber.m_f);
k_f = double(fiber.k_f);
l_f = double(fiber.l_f);
p_f = double(fiber.p_f);

% matrix
syms n_m m_m k_m l_m p_m
eqn1 = n_m == C_m(1,1);
eqn2 = l_m == C_m(1,2);
eqn3 = k_m + m_m == C_m(2,2);
```

```
eqn4 = m_m == C_m(4,4);
eqn5 = p_m == C_m(5,5);

matrix = solve([eqn1,eqn2,eqn3,eqn4,eqn5],[n_m,l_m,k_m,m_m,p_m]);
n_m = double(matrix.n_m);
m_m = double(matrix.m_m);
k_m = double(matrix.k_m);
l_m = double(matrix.l_m);
p_m = double(matrix.p_m);

% composite
k_c = (Vf*k_f*(k_m + m_m) + (1-Vf)*k_m*(k_f + m_m))/(Vf*(k_m + m_m) + (1-Vf)*(k_f +
m_m));
l_c = (Vf*l_f*(k_m + m_m) + (1-Vf)*l_m*(k_f + m_m))/(Vf*(k_m + m_m) + (1-Vf)*(k_f +
m_m));
n_c = (1-Vf)*n_m + Vf*n_f + (1-Vf)*Vf*(l_f-l_m)^2/(Vf*(k_m + m_m) + (1-Vf)*(k_f +
m_m));
m_c = (m_f*m_m*(k_m+2*m_m)+Vf*m_f*k_m*m_m+(1-Vf)*m_m^2*k_m)/(k_m*m_m+
(k_m+2*m_m)*(Vf*m_m+(1-Vf)*m_f));
p_c = p_m*(p_f*(1+Vf) + p_m*(1-Vf))/(p_f*(1-Vf) + p_m*(1+Vf));

C_c = [n_c, l_c, l_c, 0, 0, 0;
       l_c, k_c+m_c, k_c-m_c, 0, 0, 0;
       l_c, k_c-m_c, k_c+m_c, 0, 0, 0;
       0, 0, 0, m_c, 0, 0;
       0, 0, 0, 0, p_c, 0;
       0, 0, 0, 0, 0, p_c;];

S_c = inv(C_c);

%E1
E1_c = 1/S_c(1,1);
E1_c_val = subs(E1_c,Vf_val);

%E2
E2_c = 1/S_c(2,2);
E2_c_val = subs(E2_c,Vf_val);

%v12
v12_c = -S_c(1,2)*E1_c;
v12_c_val = subs(v12_c,Vf_val);

%v23
v23_c = -S_c(2,3)*E2_c;
v23_c_val = subs(v23_c,Vf_val);

%G12
G12_c = 1/S_c(5,5);
G12_c_val = subs(G12_c,Vf_val);
```

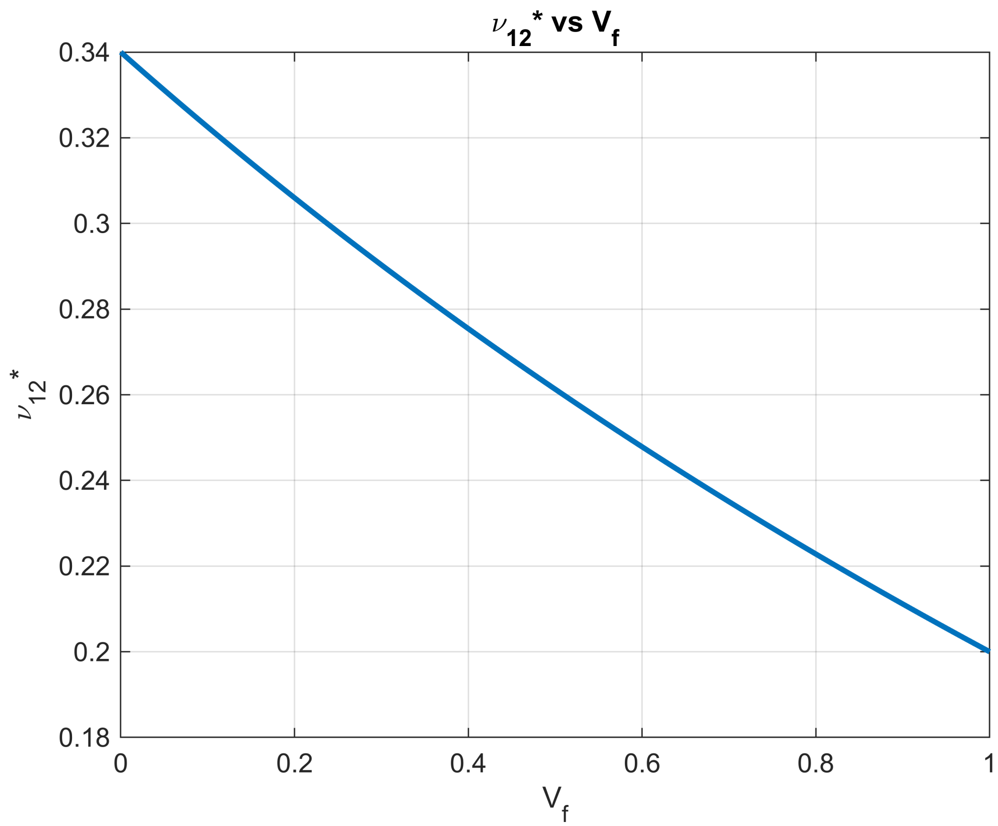
```
%G23
G23_c = 1/S_c(4,4);
G23_c_val = subs(G23_c,Vf_val);

%plot
figure;
plot(Vf_val, E1_c_val, 'LineWidth', 2);
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
grid on;
```



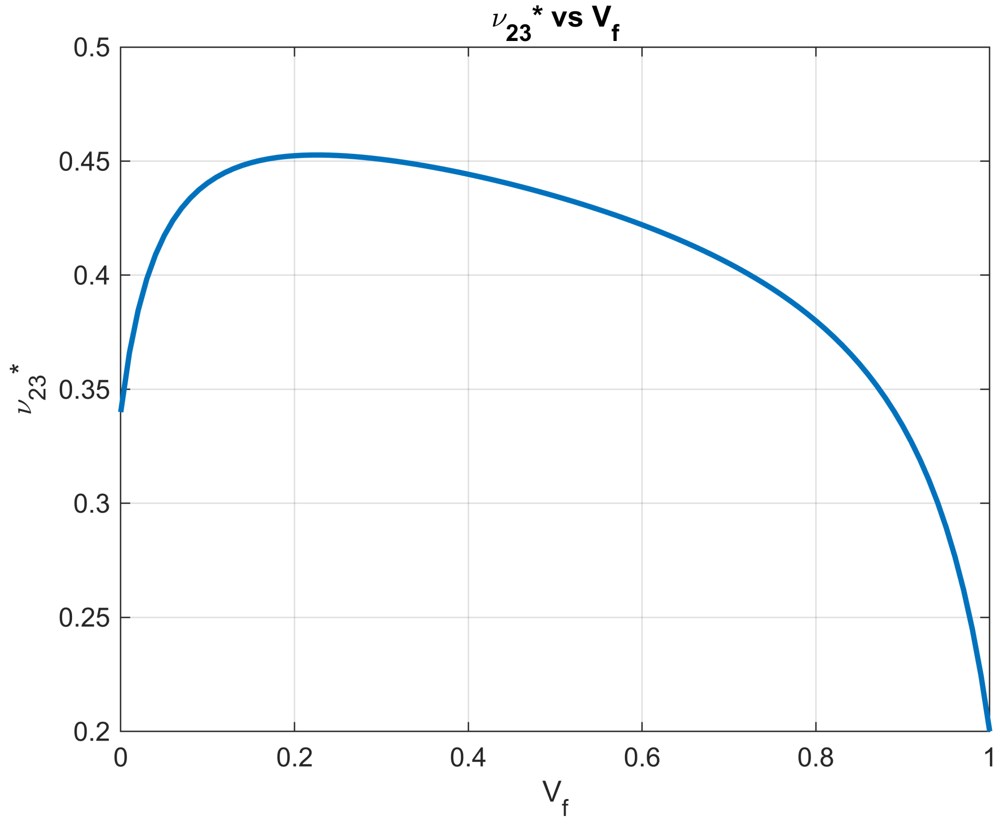

```
figure;
plot(Vf_val, E2_c_val, 'LineWidth', 2);
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
grid on;
```



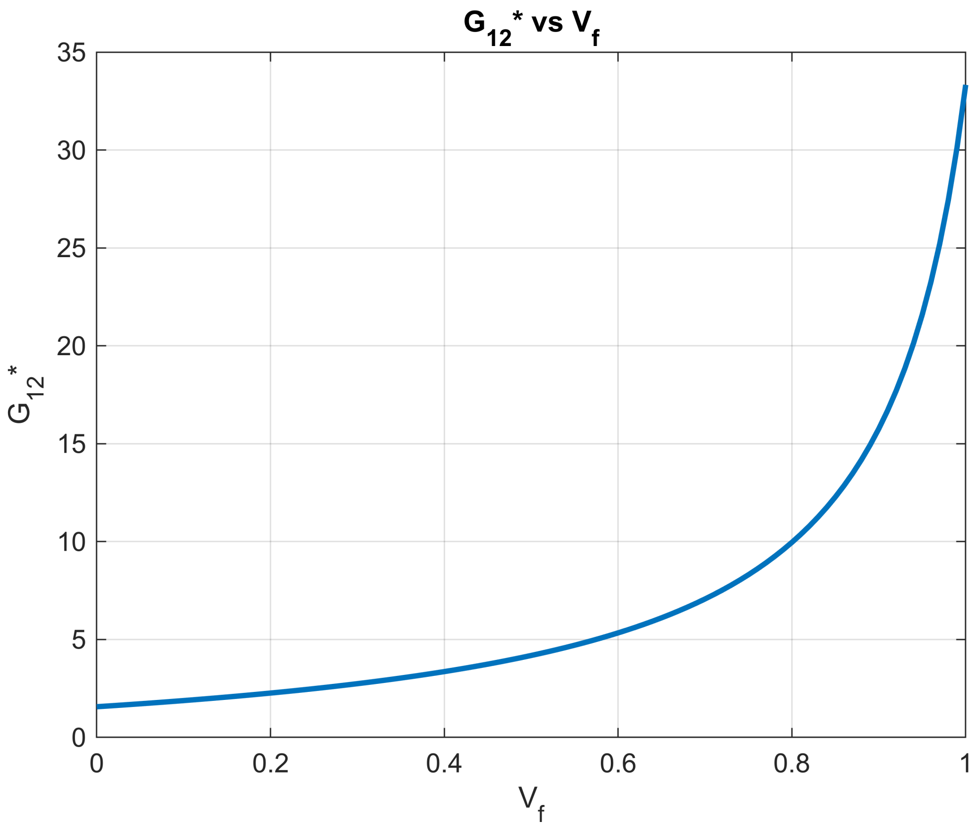

```
figure;
plot(Vf_val, v12_c_val, 'LineWidth', 2);
title('\nu_{12}* vs V_f');
xlabel('V_f'); ylabel('\nu_{12}*');
grid on;
```



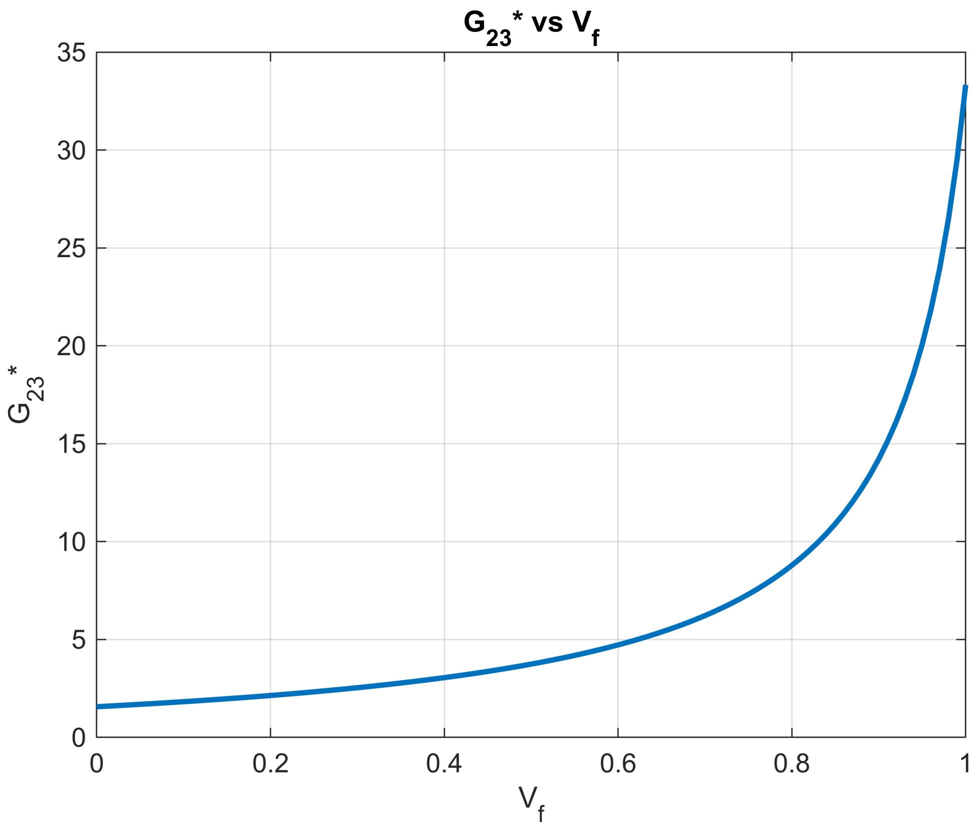

```
figure;
plot(Vf_val, v23_c_val, 'LineWidth', 2);
title('\nu_{23}* vs V_f');
xlabel('V_f'); ylabel('\nu_{23}*');
grid on;
```




```
figure;
plot(Vf_val, G12_c_val, 'LineWidth', 2);
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
grid on;
```



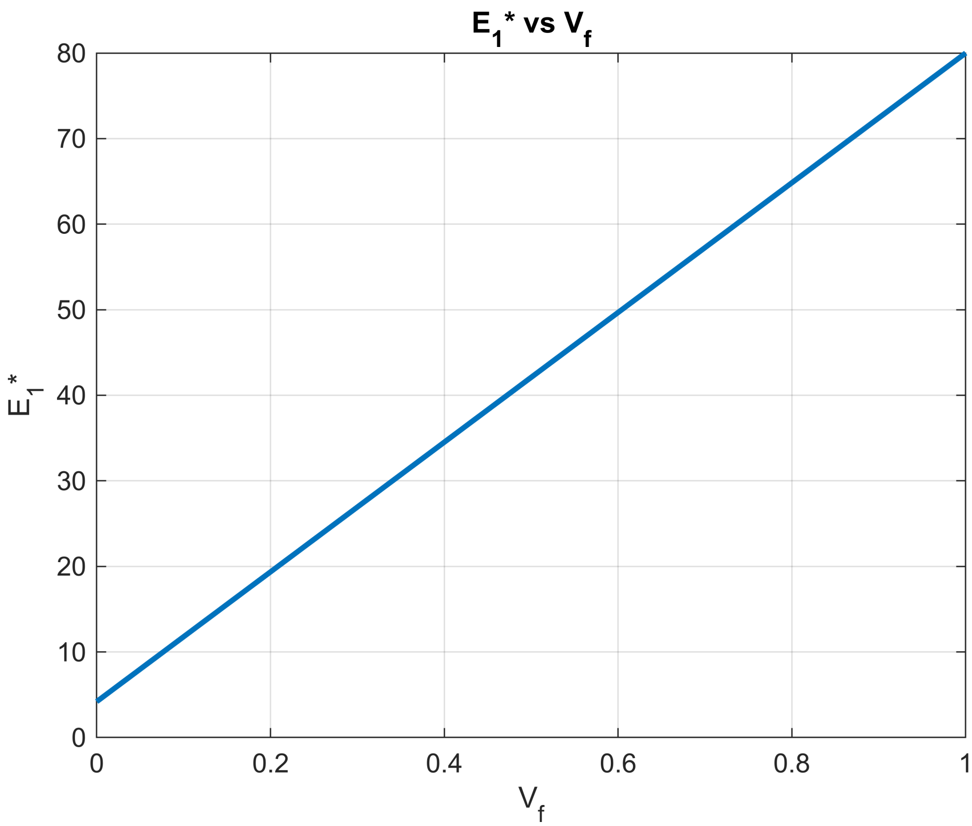

```
figure;
plot(Vf_val, G23_c_val, 'LineWidth', 2);
title('G_{23}* vs V_f');
xlabel('V_f'); ylabel('G_{23}*');
grid on;
```


G_{23}* vs V_f
35
30
25
20
15
10
5
0
0
0.2
0.4
0.6
0.8
1
G_{23}^*
V_f

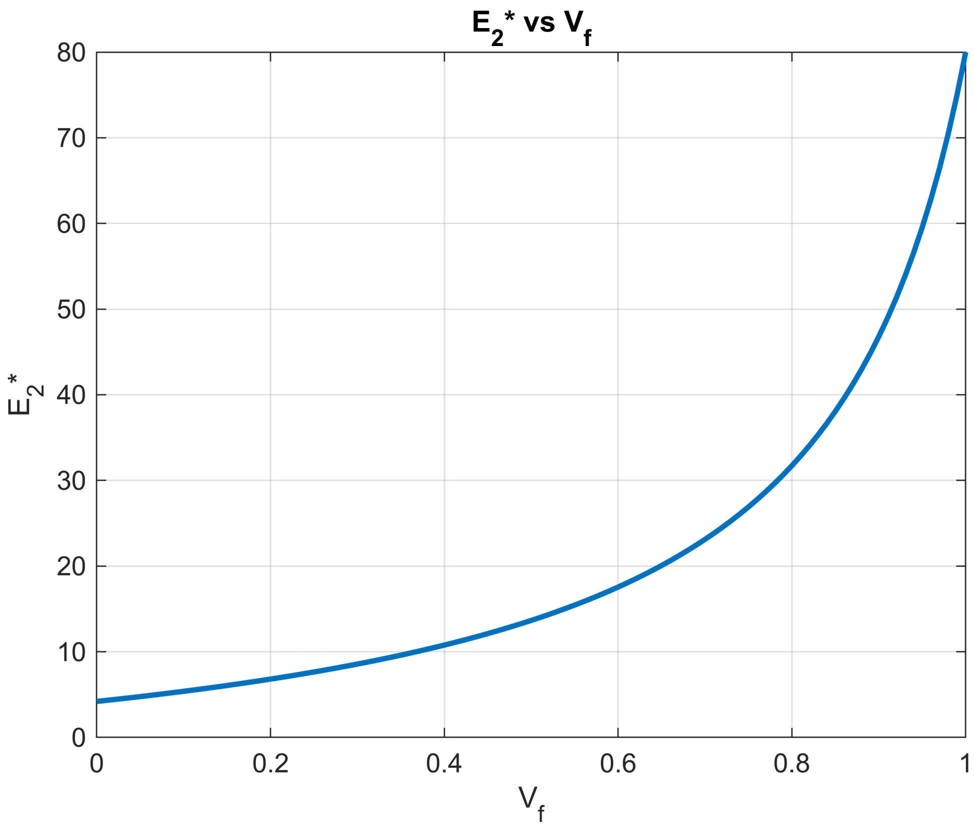
Halpin-Tsai method

```
clc;clear;
syms Vf
Vf_val = linspace(0,1,101);

E_f = 80;
G_f = 33.33;
v_f = 0.2;

E_m = 4.2;
v_m = 0.34;
G_m = E_m/(2*(1+v_m));S

zeta = 2;

eta_1 = ((E_f/E_m) - 1) / ((E_f/E_m) + zeta);
E2_c = E_m * ((1 + zeta * eta_1 * Vf) / (1 - eta_1 * Vf));

eta_2 = ((G_f/G_m) - 1) / ((G_f/G_m) + zeta);
G12_c = G_m * ((1 + zeta * eta_2 * Vf) / (1 - eta_2 * Vf));

eta_3 = ((v_f/v_m) - 1) / ((v_f/v_m) + zeta);
v23_c = v_m * ((1 + zeta * eta_3 * Vf) / (1 - eta_3 * Vf));

%E1
E1_c = Vf * E_f + (1 - Vf) * E_m;
E1_c_val = subs(E1_c,Vf_val);

%E2
E2_c_val = subs(E2_c,Vf_val);

%v12
v12_c = Vf * v_f + (1 - Vf) * v_m;
v12_c_val = subs(v12_c,Vf_val);

%v23
% q = 1 + ((4 * k_c * (v12_c)^2) / E1_c);
% v23_c = (k_c - (q * m_c)) / (k_c + (q * m_c));
% v23_c = -S_c(3,2)*E2_c;
v23_c_val = subs(v23_c,Vf_val);

%G12
G12_c_val = subs(G12_c,Vf_val);

%G23
G23_c = E2_c/(2*(1+v23_c));
G23_c_val = subs(G23_c,Vf_val);

%plotting
```

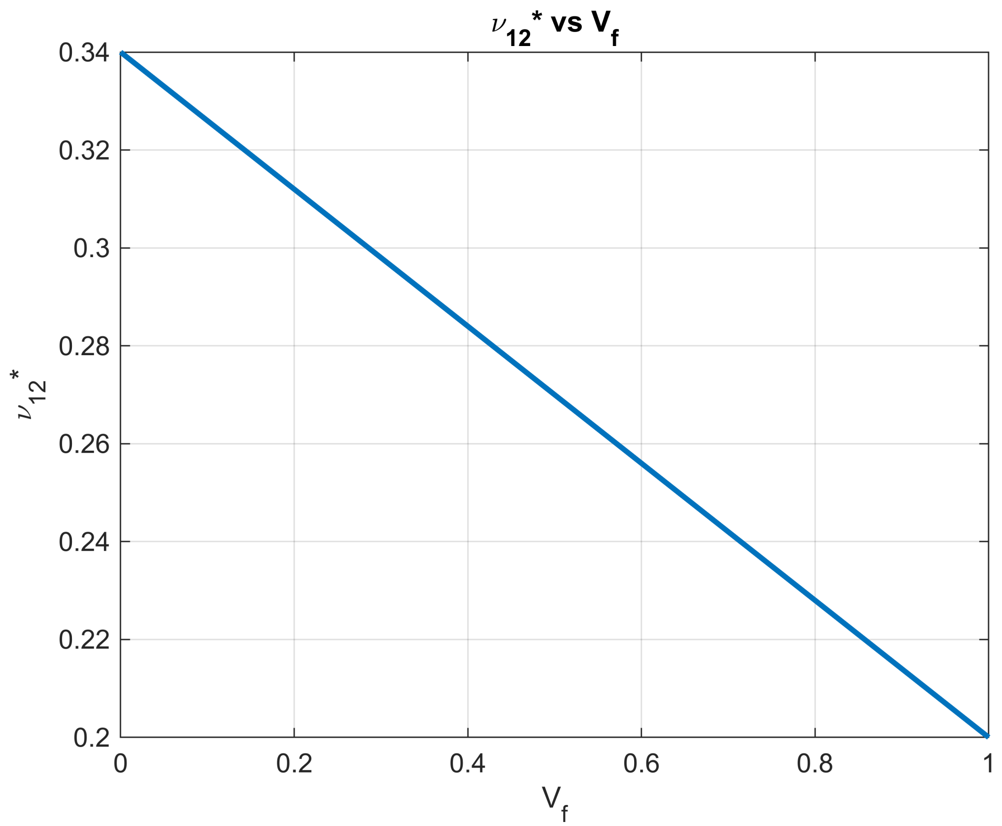
```
figure;
plot(Vf_val, E1_c_val, 'LineWidth', 2);
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
grid on;
```



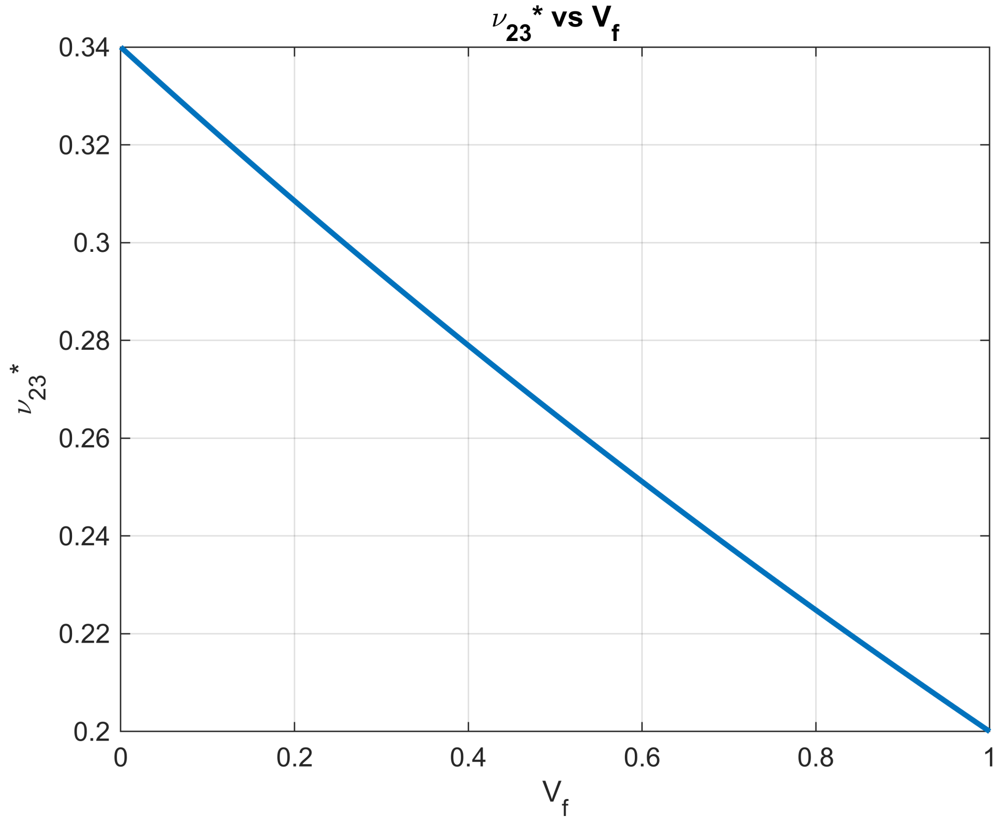

```
figure;
plot(Vf_val, E2_c_val, 'LineWidth', 2);
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
grid on;
```



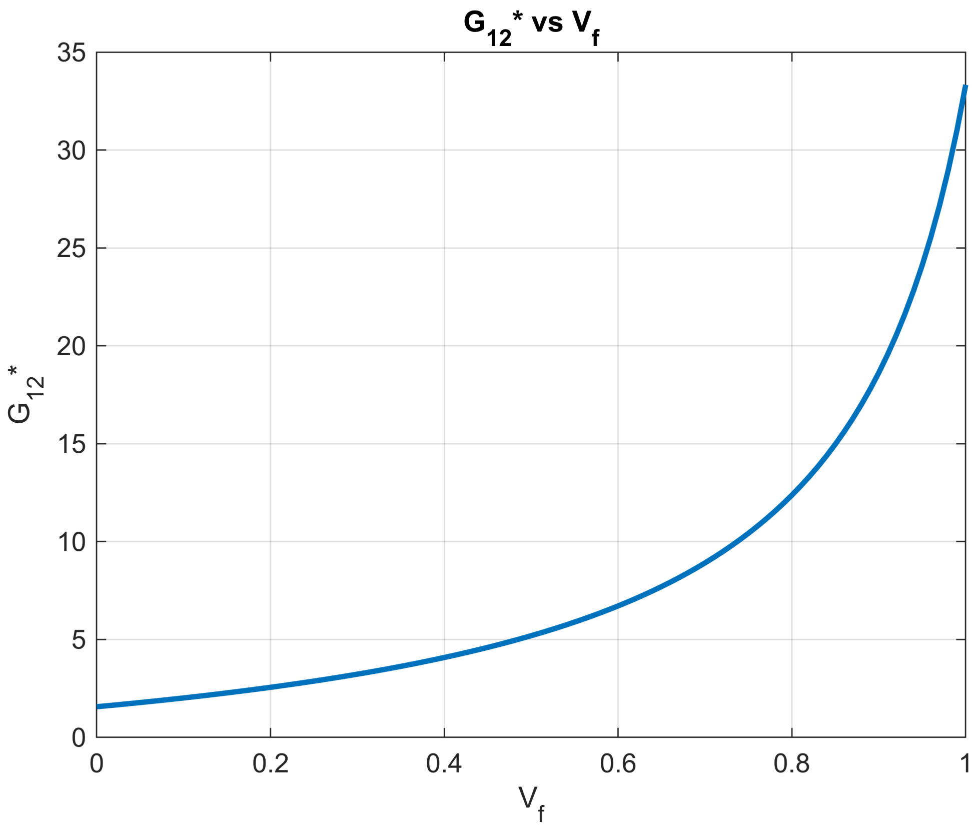

```
figure;
plot(Vf_val, v12_c_val, 'LineWidth', 2);
title('\nu_{12}* vs V_f');
xlabel('V_f'); ylabel('\nu_{12}*');
grid on;
```



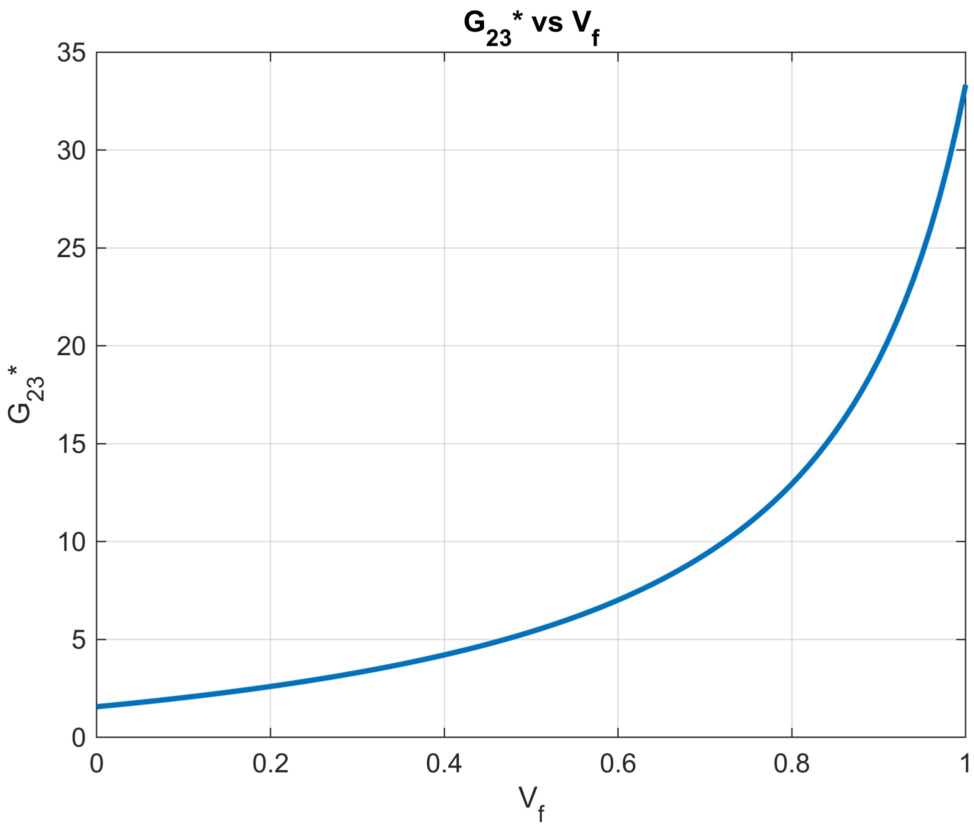

```
figure;
plot(Vf_val, v23_c_val, 'LineWidth', 2);
title('\nu_{23}* vs V_f');
xlabel('V_f'); ylabel('\nu_{23}*');
grid on;
```




```
figure;
plot(Vf_val, G12_c_val, 'LineWidth', 2);
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
grid on;
```




```
figure;
plot(Vf_val, G23_c_val, 'LineWidth', 2);
title('G_{23}* vs V_f');
xlabel('V_f'); ylabel('G_{23}*');
grid on;
```

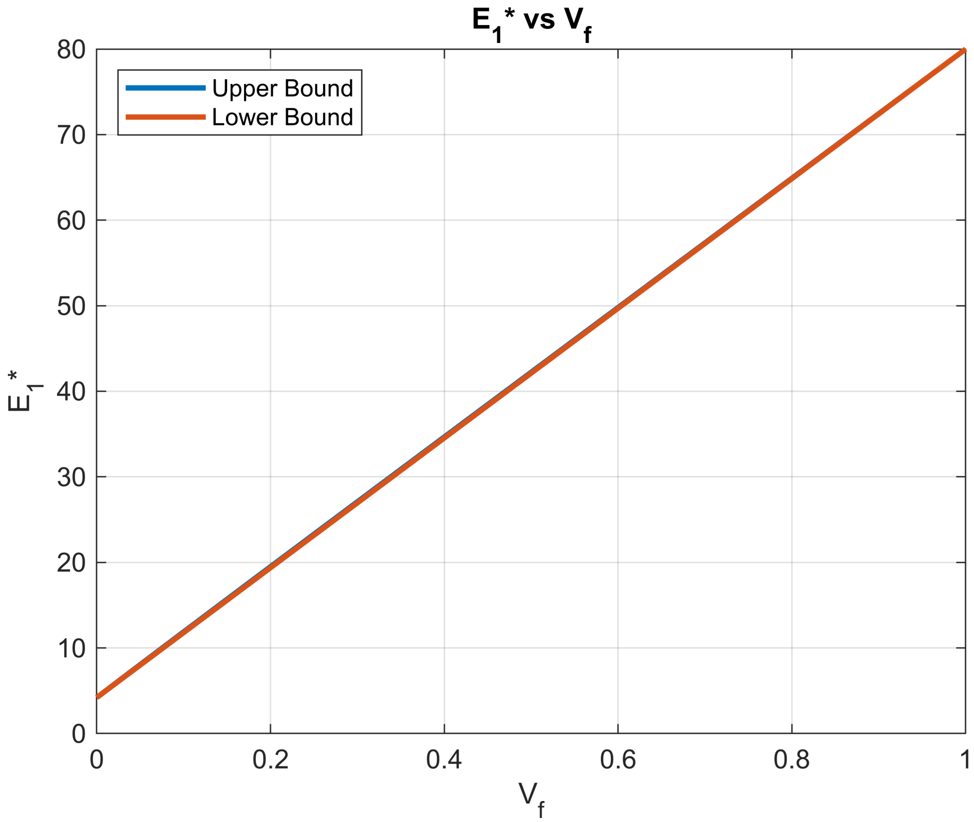


G23* vs Vf
35
30
25
20
15
10
5
0
G23*
0
0.2
0.4
0.6
0.8
1
Vf

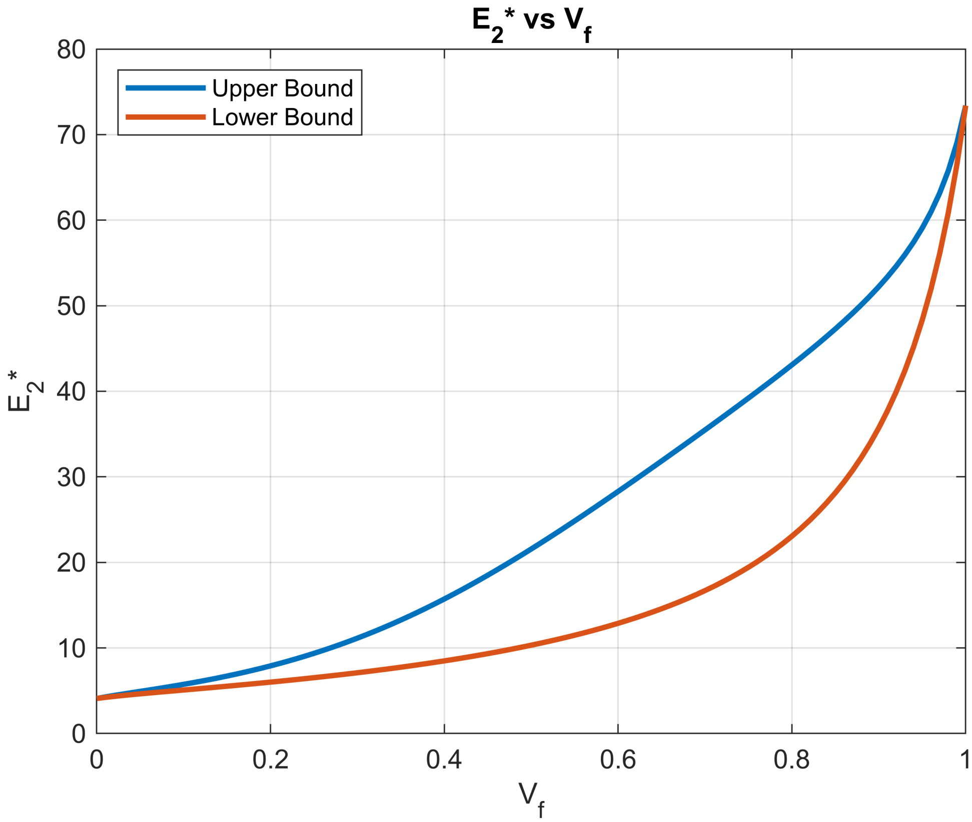
Hashin Shtrikman Bounds

```
clc;clear;
syms Vf
Vf_val = linspace(0,1,101);

E_f = 80;
G_f = 33.33;
v_f = 0.2;
K_f = E_f/(3*(1-2*v_f));

E_m = 4.2;
v_m = 0.34;
a_m = 45;
G_m = E_m/(2*(1+v_m));
K_m = E_m/(3*(1-2*v_m));

%E1
E1_minus = (E_m * (1-Vf)) + (E_f * Vf) + ((4 * ((v_f - v_m)^2) * Vf * (1-Vf)) /
(((1-Vf) / K_f) + (Vf / K_m) + (1 / G_m)));
E1_minus = subs(E1_minus,Vf_val);
E1_plus = (E_m * (1-Vf)) + (E_f * Vf) + ((4 * ((v_f - v_m)^2) * Vf * (1-Vf)) /
(((1-Vf) / K_f) + (Vf / K_m) + (1 / G_f)));
E1_plus = subs(E1_plus,Vf_val);

%v12
v12_minus = v_m * (1-Vf) + v_f * Vf + ( (v_m - v_f) * (1 / K_f - 1 / K_m) * Vf *
(1-Vf)) / ((1-Vf) / K_f + Vf / K_m + 1 / G_f);
v12_minus = subs(v12_minus,Vf_val);
v12_plus = v_m * (1-Vf) + v_f * Vf + ( (v_f - v_m) * (1 / K_m - 1 / K_f) * Vf *
(1-Vf)) / ((1-Vf) / K_f + Vf / K_m + 1 / G_m);
v12_plus = subs(v12_plus,Vf_val);

%G12
G12_minus = G_m * ((G_m * (1-Vf) + G_f * (1 + Vf)) / (G_f * (1-Vf) + G_m * (1 +
Vf)));
G12_minus = subs(G12_minus,Vf_val);
G12_plus = G_f * ((G_f * Vf + G_m * (1 + (1-Vf))) / (G_m * Vf + G_f * (1 +
(1-Vf))));
G12_plus = subs(G12_plus,Vf_val);

%constant
gamma = G_f / G_m;
beta1 = K_f / (K_f + (2 * G_m));
beta2 = K_f / (K_f + (2 * G_f));
alpha = (beta1 - gamma * beta2) / (1 + gamma * beta2);
rho = (gamma + beta1) / (gamma - 1);

%G23
```

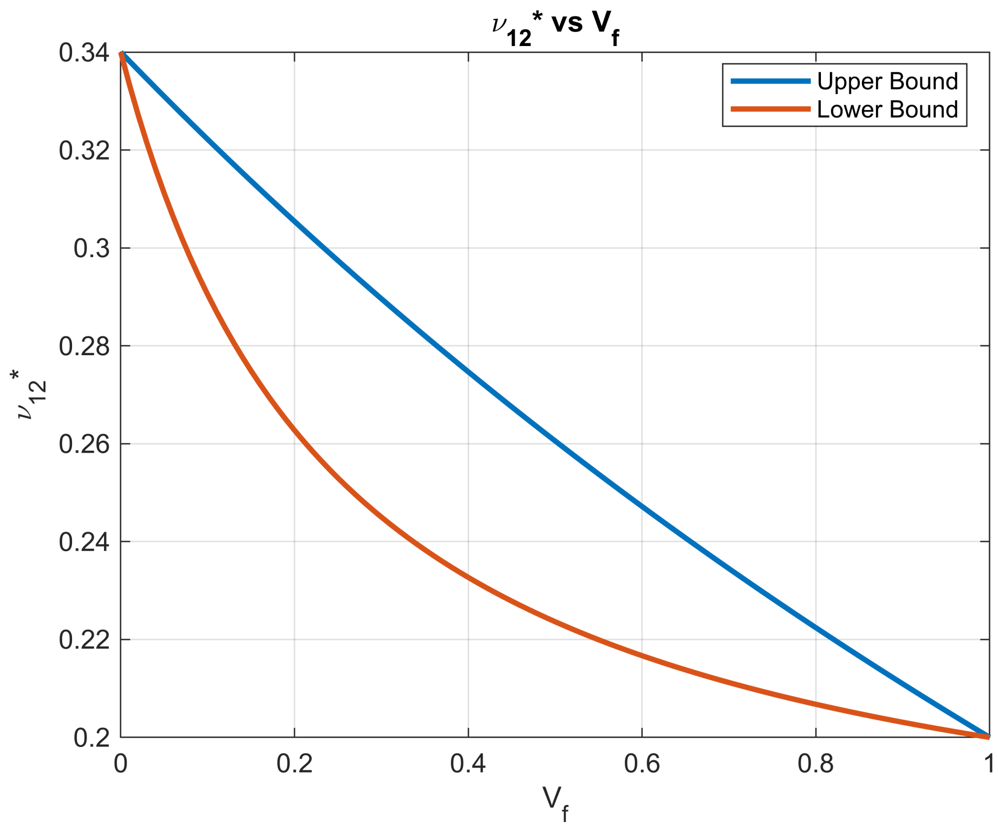
```
G23_minus = G_m + (Vf / (1 / (G_f - G_m) + ((K_m + 2 * G_m) * (1-Vf)) / (2 * G_m *
(K_m + G_m))));
G23_minus = subs(G23_minus,Vf_val);
G23_plus = G_m * ( 1 + (((1 + beta1) * Vf) / (rho - (Vf * (1 + ((3 * beta1^2 *
(1-Vf)^2) / (alpha * (Vf^3) + 1)))))));
G23_plus = subs(G23_plus,Vf_val);

%K
K_minus = K_m + (Vf / (1 / (K_f - K_m) + ((1-Vf) / (K_m + G_m))));
K_minus = subs(K_minus,Vf_val);
K_plus = K_f + ((1-Vf) / (1 / (K_m - K_f) + (Vf / (K_f + G_f))));
K_plus = subs(K_plus,Vf_val);

%E2
E2_minus = (4 * G23_minus .* K_minus) ./ (K_minus + G23_minus + ((4 * G23_minus .*
K_minus .* (v12_plus).^2) ./ E1_minus));
E2_plus = (4 * G23_plus .* K_plus) ./ (K_plus + G23_plus + ((4 * G23_plus .*
K_plus .* (v12_minus).^2) ./ E1_plus));

%v23
v23_minus = (K_minus - G23_plus - (4 * G23_plus .* K_minus .* (v12_plus).^2) ./
E1_minus) ./ (K_minus + G23_plus + (4 * G23_plus .* K_minus .* (v12_plus).^2) ./
E1_minus);
v23_plus = (K_plus - G23_minus - (4 * G23_minus .* K_plus .* (v12_minus).^2) ./
E1_plus) ./ (K_plus + G23_minus + (4 * G23_minus .* K_plus .* (v12_minus).^2) ./
E1_plus);

%plotting
plot(Vf_val,E1_plus, 'LineWidth', 2)
hold on;
plot(Vf_val,E1_minus, 'LineWidth', 2)
title('E_1* vs V_f');
xlabel('V_f'); ylabel('E_1*');
legend('Upper Bound', 'Lower Bound', 'Location', 'northwest')
grid on;
hold off;
```



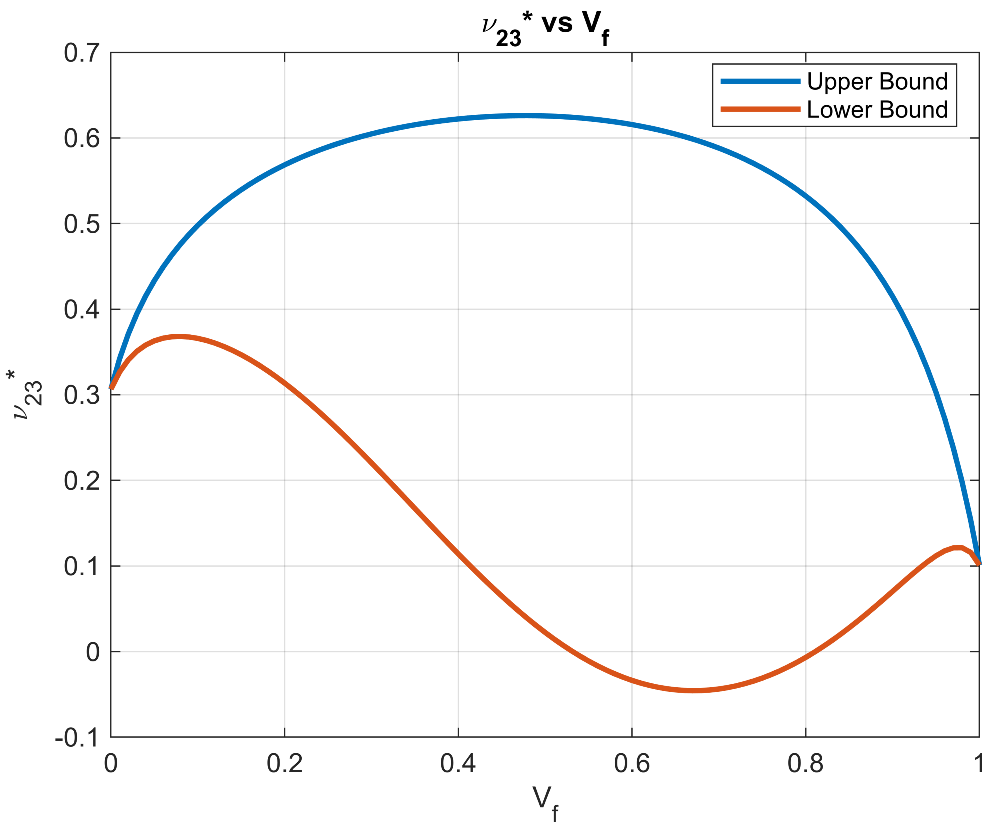

```
plot(Vf_val, E2_plus, 'LineWidth', 2)
hold on;
plot(Vf_val, E2_minus, 'LineWidth', 2)
title('E_2* vs V_f');
xlabel('V_f'); ylabel('E_2*');
legend('Upper Bound', 'Lower Bound', 'Location', 'northwest')
grid on
hold off;
```



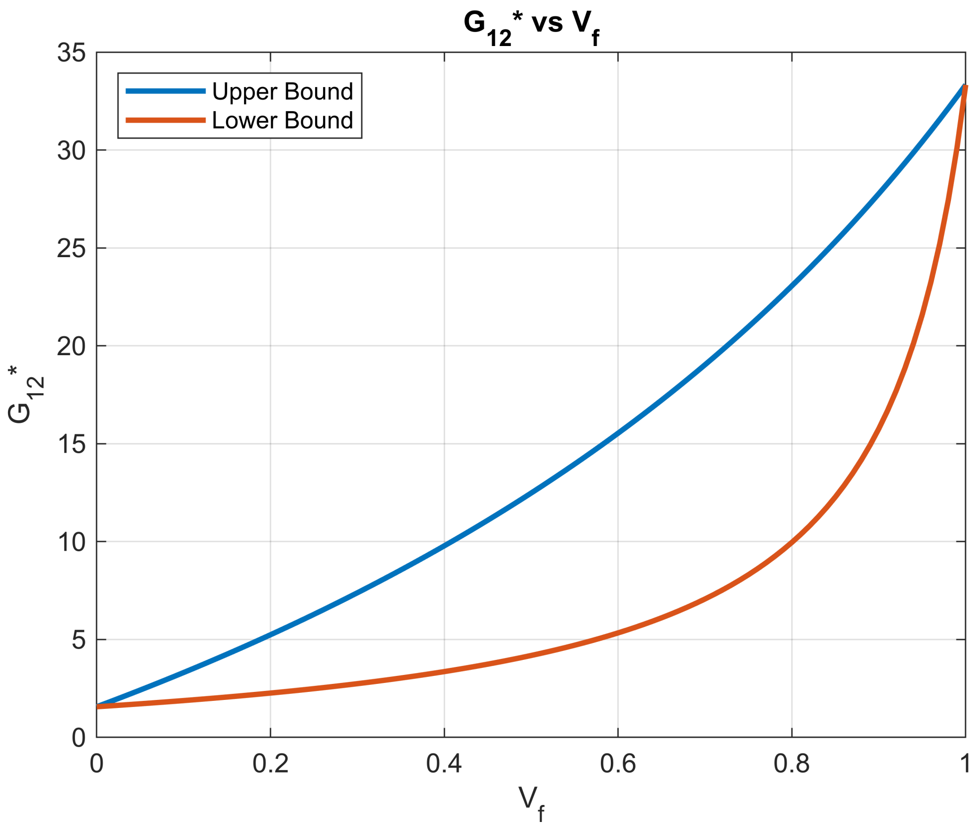

```
plot(Vf_val,v12_plus, 'LineWidth', 2)
hold on;
plot(Vf_val,v12_minus, 'LineWidth', 2)
title('\nu_{12}* vs V_f');
xlabel('V_f'); ylabel('\nu_{12}*');
legend('Upper Bound', 'Lower Bound', 'Location', 'best')
grid on
hold off;
```



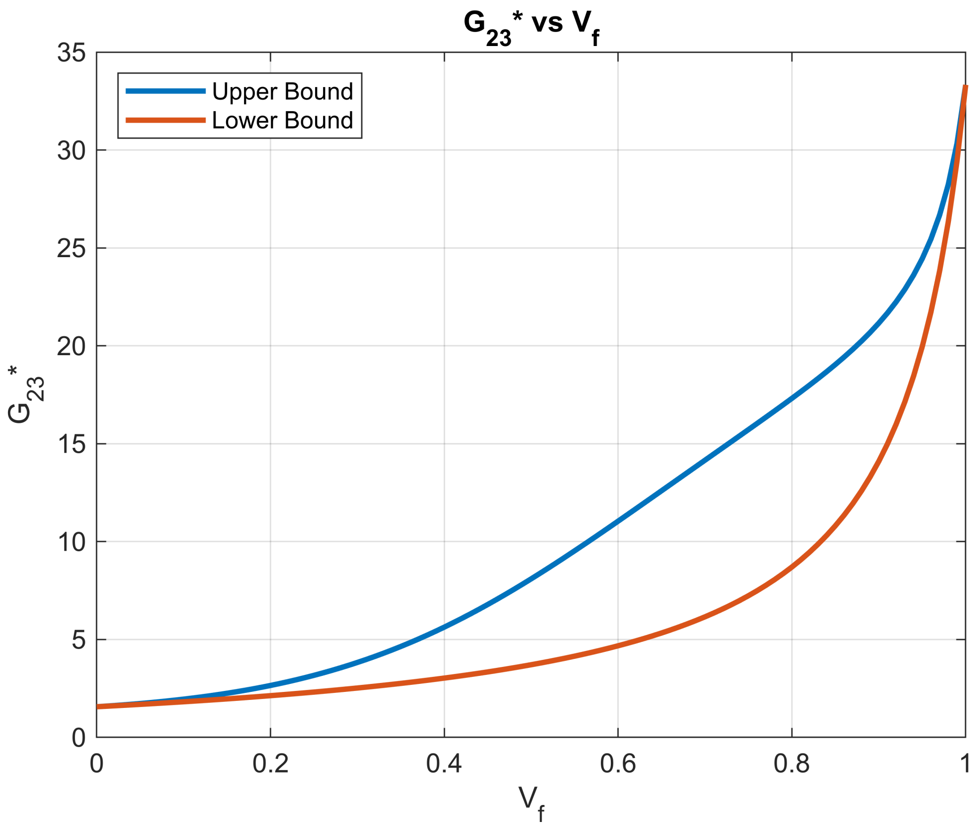

```
figure;
plot(Vf_val, v23_plus, 'LineWidth', 2)
hold on;
plot(Vf_val, v23_minus, 'LineWidth', 2)
title('\nu_{23}* vs V_f');
xlabel('V_f'); ylabel('\nu_{23}*');
legend('Upper Bound', 'Lower Bound', 'Location', 'best')
grid on
hold off;
```



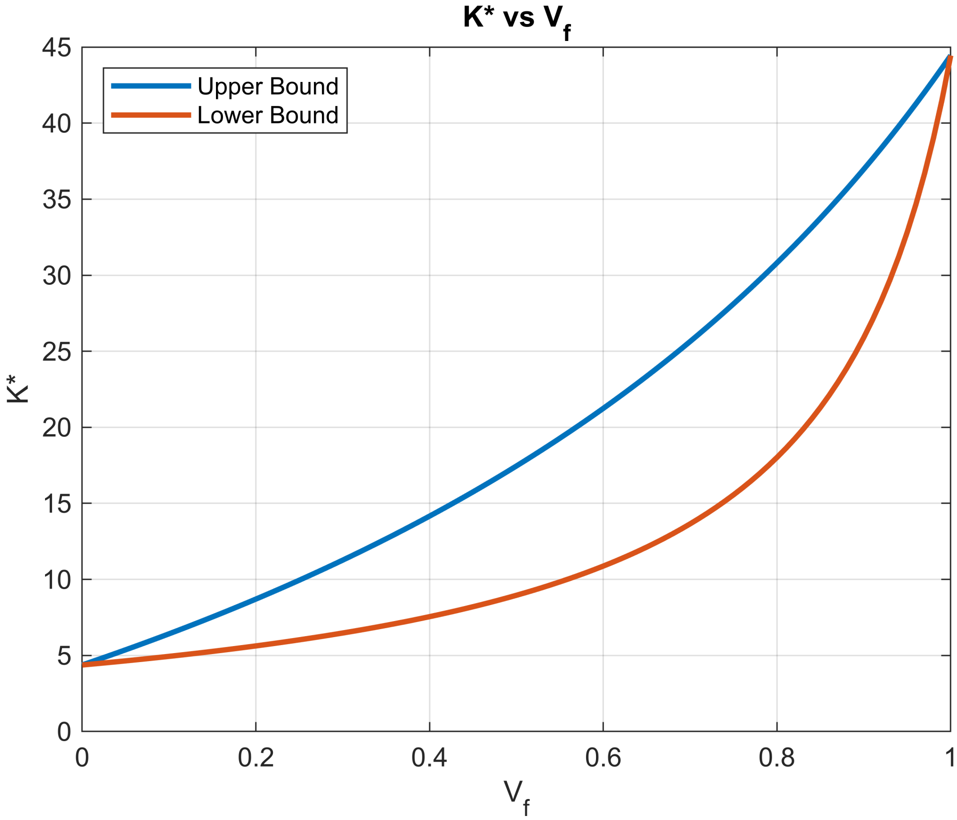

```
plot(Vf_val,G12_plus, 'LineWidth', 2)
hold on;
plot(Vf_val,G12_minus, 'LineWidth', 2)
title('G_{12}* vs V_f');
xlabel('V_f'); ylabel('G_{12}*');
legend('Upper Bound', 'Lower Bound', 'Location', 'northwest')
grid on
hold off;
```




```
plot(Vf_val,G23_plus, 'LineWidth', 2)
hold on;
plot(Vf_val,G23_minus, 'LineWidth', 2)
title('G_{23}* vs V_f');
xlabel('V_f'); ylabel('G_{23}*');
legend('Upper Bound', 'Lower Bound', 'Location', 'northwest')
grid on
hold off;
```




```
plot(Vf_val,K_plus, 'LineWidth', 2)
hold on;
plot(Vf_val,K_minus, 'LineWidth', 2)
title('K* vs V_f');
xlabel('V_f'); ylabel('K*');
legend('Upper Bound', 'Lower Bound', 'Location', 'northwest')
grid on
hold off;
```


K* vs V_f
Upper Bound
Lower Bound
K*
V_f
0
5
10
15
20
25
30
35
40
45
0
0.2
0.4
0.6
0.8
1