

Stress free condition => Reverse Martensitic Transformation

With:

**Code File:**

*import math*

*import numpy as np*

*import matplotlib.pyplot as plt*

*from mpl\_toolkits.mplot3d import Axes3D*

*class Shape\_Memory\_Effect:*

*def \_\_init\_\_(self, YA, YM, Ms, Mf, As, Af, CM, CA, TCRS, TCRF, SL, temp, shiS0, shiT0):*

*self.YA = YA*

*self.YM = YM*

*self.Ms = Ms*

*self.Mf = Mf*

*self.As = As*

*self.Af = Af*

*self.CM = CM*

*self.CA = CA*

*self.TCRS = TCRS*

*self.TCRF = TCRF*

*self.SL = SL*

*self.temp = temp*

*self.shiS0 = shiS0*

*self.shiT0 = shiT0*

*self.shi\_lst = []*

*def stress\_calculated(self):*

*global Ta, Tb, Tc*

*Ta = 0*

*Tb = self.TCRS*

*Tc = self.TCRF*

*def predicted\_through\_stress(self):*

*global Tb\_predicted, Tc\_predicted, Td\_predicted, Taf\_predicted, Tkf\_predicted, Ta\_predicted*

*Tb\_predicted = np.linspace(Ta, Tb, 20)*

*Tc\_predicted = np.linspace(Tb, Tc, 20)*

*for i in Tc\_predicted:*

*shiS = ((1 - self.shiS0) / 2) \* math.cos((math.pi / (self.TCRS - self.TCRF)) \* (i - self.TCRF)) + ((1 + self.shiS0) / 2)*

*self.shi\_lst.append(shiS)*

*self.shi\_lst = np.array(self.shi\_lst)*

*Td\_predicted = np.linspace(Tc,Ta,num=20, endpoint=True)*

*Taf\_predicted = np.linspace(0,0, num=20, endpoint=True)*

*Tkf\_predicted = np.linspace(0,0, num=20, endpoint=True)*

*Ta\_predicted = np.linspace(0,0, num=20, endpoint=True)*

*def predicted\_through\_strain(self):*

*global Sb\_predicted, Sc\_predicted, Sd\_predicted*

*Sb\_predicted = Tb\_predicted/self.YM*

*Sc\_predicted = (Tc\_predicted/self.YM) + self.SL\*self.shi\_lst*

*Sd\_predicted = (Td\_predicted/self.YM) + self.SL*

*def predicted\_through\_temp(self):*

*global temp\_b\_predicted, temp\_c\_predicted, temp\_d\_predicted, temp\_a\_f\_predicted, temp\_k\_f\_predicted, temp\_a\_predicted*

*temp\_b\_predicted = np.linspace(self.temp,self.temp,num=20, endpoint=True)*

*temp\_c\_predicted = np.linspace(self.temp,self.temp, num=20, endpoint=True)*

*temp\_d\_predicted = np.linspace(self.temp,self.temp, num=20, endpoint=True)*

*temp\_a\_f\_predicted = np.linspace(self.temp, self.As, num=20, endpoint=True)*

*temp\_k\_f\_predicted = np.linspace(self.As, self.Af, num=20, endpoint=True)*

*temp\_a\_predicted = np.linspace(self.Af, self.temp, num=20, endpoint=True)*

*def predicted\_through\_strain\_by\_temp(self):*

*global strain\_a\_f\_predicted, strain\_k\_f\_predicted, strain\_a\_predicted*

*# At a'*

*strain\_a\_f\_predicted = np.linspace(self.SL,self.SL, num=20, endpoint=True)*

*# At k'*

*shi\_M\_ls = []*

*for i in temp\_k\_f\_predicted:*

*shi\_M = 1/2\*(1 + math.cos((math.pi/(self.Af-self.As))\*(i-self.As)))*

*shi\_M\_ls.append(shi\_M)*

*shi\_M\_ls = np.array(shi\_M\_ls)*

*strain\_k\_f\_predicted = self.SL\*shi\_M\_ls*

*# At a*

*strain\_a\_predicted = np.linspace(0,0, num=20, endpoint=True)*

*def plot\_T\_S\_Temp\_curve(self):*

*strain\_combined = np.concatenate((Sb\_predicted, Sc\_predicted, Sd\_predicted, strain\_a\_f\_predicted, strain\_k\_f\_predicted, strain\_a\_predicted))*

*stress\_combined = np.concatenate((Tb\_predicted, Tc\_predicted, Td\_predicted, Taf\_predicted, Tkf\_predicted, Ta\_predicted))*

*temp\_combined = np.concatenate((temp\_b\_predicted, temp\_c\_predicted, temp\_d\_predicted, temp\_a\_f\_predicted, temp\_k\_f\_predicted, temp\_a\_predicted))*

*fig = plt.figure(figsize=(12, 8))*

*ax = fig.add\_subplot(111, projection='3d')*

*ax.plot(temp\_combined, strain\_combined, stress\_combined, label='Stress-Strain-Temp Curve', color='blue')*

*ax.set\_xlabel('Temperature (°C)')*

*ax.set\_ylabel('Strain')*

*ax.set\_zlabel('Stress (MPa)')*

*ax.set\_title('Stress-Strain-Temperature Relationship')*

*ax.set\_zlim(0, 170)*

*ax.legend()*

*ax.grid(True)*

*plt.show()*

*def main():*

*Mf = 9  # °C*

*Ms = 18  # °C*

*As = 35  # °C*

*Af = 49  # °C*

*CM = 8  # MPa/°C (slope for martensite)*

*CA = 14  # MPa/°C (slope for austenite)*

*YM = 26000*

*YA = 67000*

*TCRS = 100  # MPa (start transformation stress)*

*TCRF = 170  # MPa (finish transformation stress)*

*shiT0 = 1*

*shiS0 = 0*

*SL = 0.07*

*temp = 5*

*proc = Shape\_Memory\_Effect(YA, YM, Ms, Mf, As, Af, CM, CA, TCRS, TCRF, SL, temp, shiS0, shiT0)*

*proc.stress\_calculated()*

*proc.predicted\_through\_stress()*

*proc.predicted\_through\_strain()*

*proc.predicted\_through\_temp()*

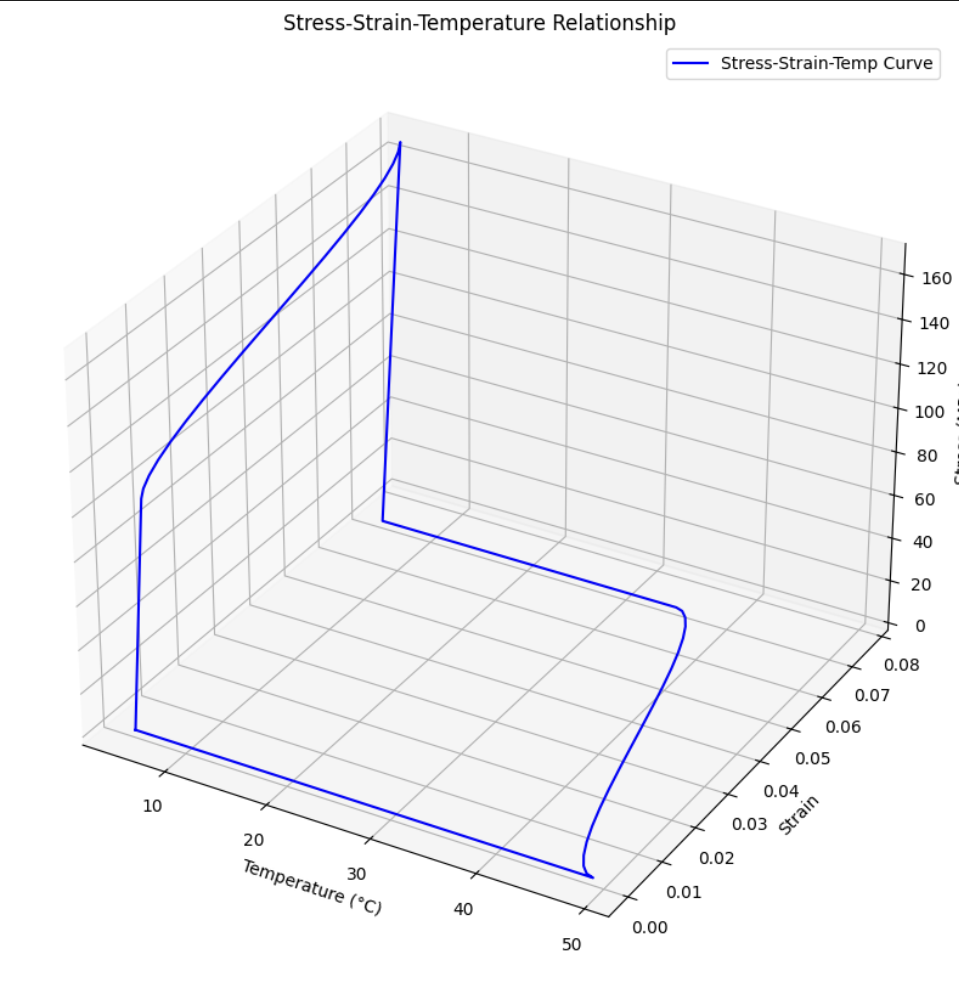
*proc.predicted\_through\_strain\_by\_temp()*

*proc.plot\_T\_S\_Temp\_curve()*

*if \_\_name\_\_ == "\_\_main\_\_":*

*main()*

**Result:**

****