problem_set_5

November 1, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import pandas as pd
```

0.0.1 Question 1

part A & B

[2]:
```
muA = -1 #kJ/mol
muB = -15 #kJ/mol
dG_st = muB - muA #KJ/mol
dG_st *= 1000 #J / mol
R = 8.3145 #J/ mol K
T = 310 #K
Keq = np.e ** (-dG_st/(R*T))
print('deltaG Standard (Joules/mol): ', dG_st)
print('Keq: ', Keq)
```

```
deltaG Standard (Joules/mol):  -14000
Keq:  228.5215058109785
```

Part C

[3]:
```
B = 0.01 #mol/1L
A = 1.99 #mol/1L
Q = B/A

dG = dG_st + R*T*np.log(Q)
print('deltaG (Joules/mol): ', dG)
```

```
deltaG (Joules/mol):  -27643.466719203258
```

Part D

[4]:
```
n_tot = 2 #mol
L = 1 #Liters
nA = 2/(Keq + 1) #mol
nB = n_tot - nA #mol
M_A = nA/L
```

```python
M_B = nB/L
print(f'[A] = {M_A:.4f} M and [B] = {M_B:.4f} M')
```

```
[A] = 0.0087 M and [B] = 1.9913 M
```

Part E

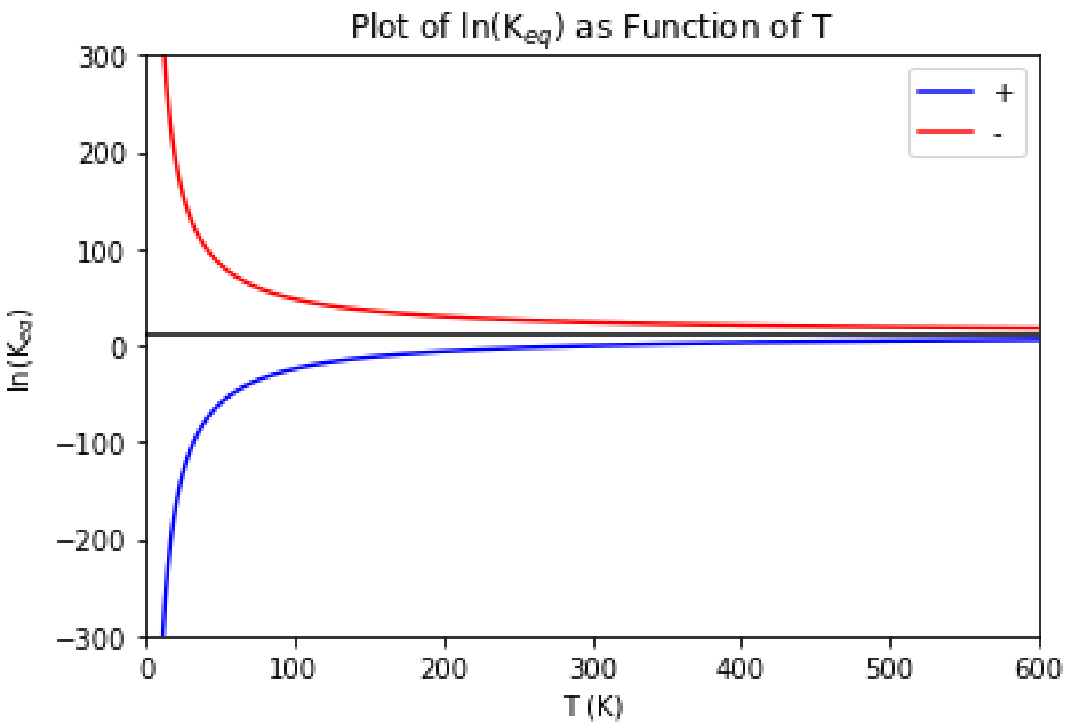
[5]:
```python
Gt_I = 1*muA*1000 + 1*muB*1000
Gt_F = nA*(muA*1000 + R*T*np.log(nA/1)) + nB*(muB*1000 + R*T*np.log(nB))

DGt = Gt_F - Gt_I
print(Gt_I)
print(Gt_F)
print('',DGt)
```

```
-16000
-26449.34201097052
 -10449.342010970518
```

0.0.2 Question 2

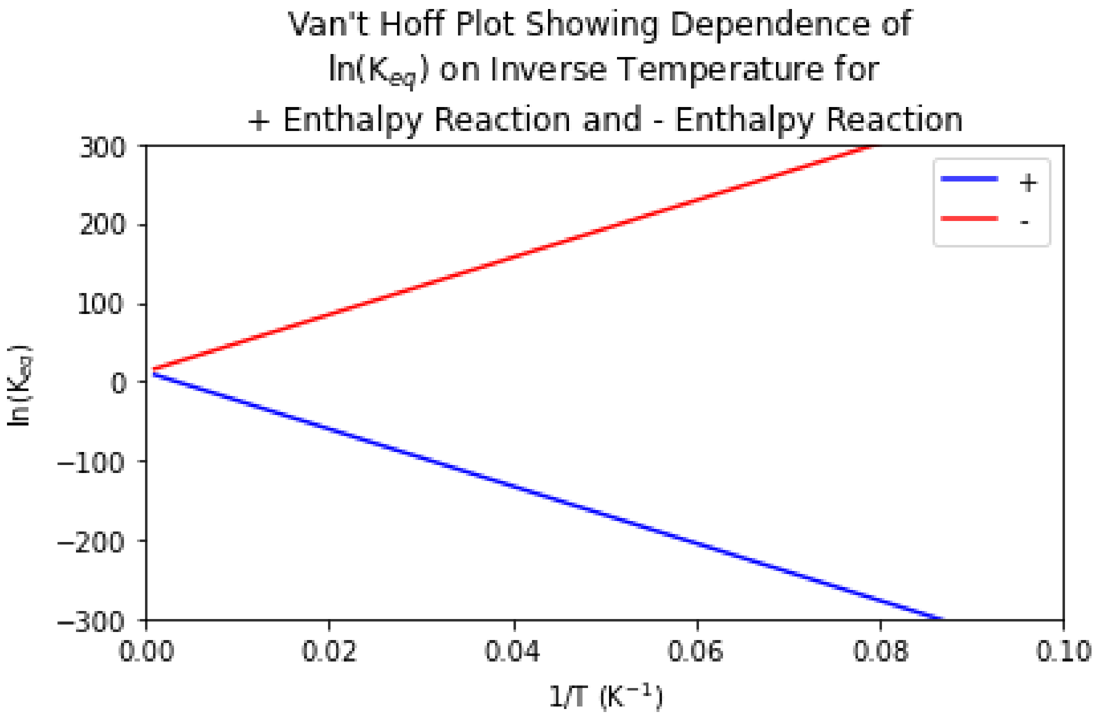
Part A

[6]:
```python
DSst = 100 #J/K mol
def ln(T, DH_st, DS_st):
    R = 8.3145 #J/mol K
    return -DH_st/(R*T) + DS_st/R

X = np.arange(1, 601, 1)
positive = ln(X, 30000, DSst)
negative = ln(X, -30000, DSst)

plt.plot(X, positive, 'b', label="+")
plt.plot(X, negative,'r', label="-")
plt.axis([0, 600, -300, 300])
plt.legend()
plt.xlabel("T (K)")
plt.ylabel("ln(K$_{eq}$)")
plt.title("Plot of ln(K$_{eq}$) as Function of T")
plt.hlines(DSst/R, 0, 1000,'k');
```



Part F
> The slope of the Van't Hoff plot is equal to -ΔH / R and thus is always linear. It is positive when ΔH is negative and negative when ΔH is positive.

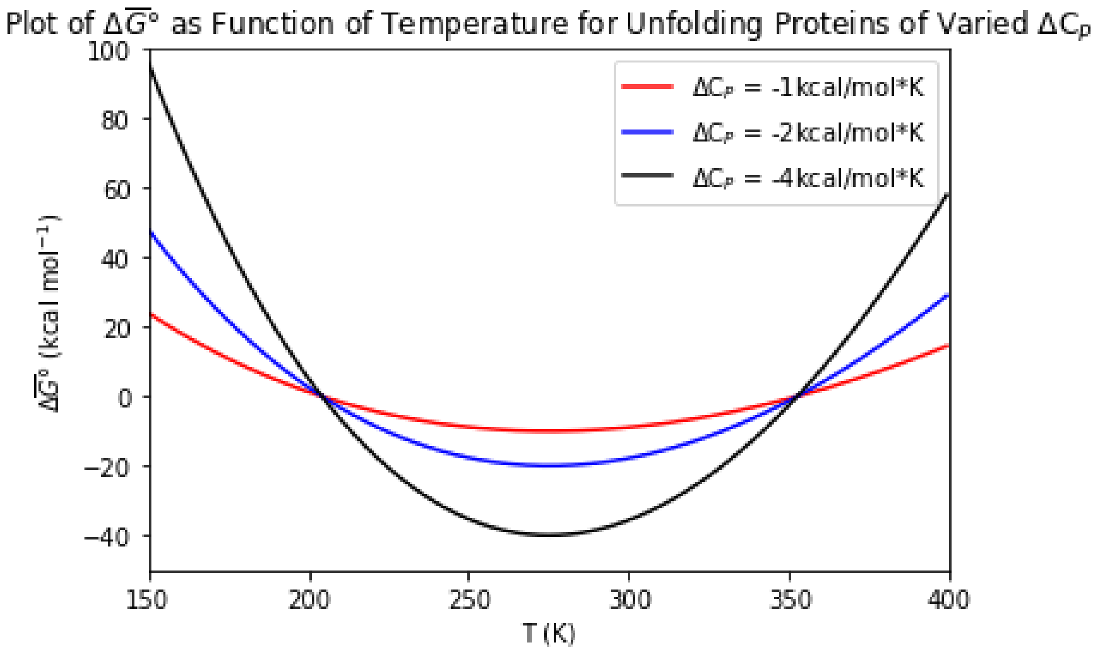
[7]:
```
Xs = np.arange(0.001, 0.1001, 0.001)
vh_Xs = np.array([1/x for x in Xs])

pos = ln(vh_Xs, 30000, DSst)
neg = ln(vh_Xs, -30000, DSst)

plt.plot(Xs, pos, "b", label="+")
plt.plot(Xs, neg, "r", label="-")
plt.legend()
plt.axis([0,0.1,-300,300])
plt.xlabel("1/T (K$^{-1}$)")
plt.ylabel("ln(K$_{eq}$)")
plt.title("Van't Hoff Plot Showing Dependence of \nln(K$_{eq}$) on Inverse␣
↪Temperature for \n+ Enthalpy Reaction and - Enthalpy Reaction")
plt.tight_layout();
```



0.0.3 Problem 5.4

Part A

[8]:
```
def DGbarstd(T, DCp, Th=265, Ts=275):
    return DCp*(T-Th) - T*DCp*np.log(T/Ts)

Ts = np.arange(150, 400, 1)
A = DGbarstd(Ts, -1)
B = DGbarstd(Ts, -2)
C = DGbarstd(Ts, -4)

plt.plot(Ts, A,'r', label="$\Delta$C$_P$ = -1kcal/mol*K")
plt.plot(Ts, B,'b', label="$\Delta$C$_P$ = -2kcal/mol*K")
plt.plot(Ts, C,'k', label="$\Delta$C$_P$ = -4kcal/mol*K")
plt.legend()
plt.axis([150, 400, -50, 100])
plt.xlabel("T (K)")
plt.title("Plot of $\Delta$$\overline{G}$$\degree$ as Function of Temperature␣
↪for Unfolding Proteins of Varied $\Delta$C$_P$")
plt.ylabel("$\Delta$$\overline{G}$$\degree$ (kcal mol$^{-1}$)");
```

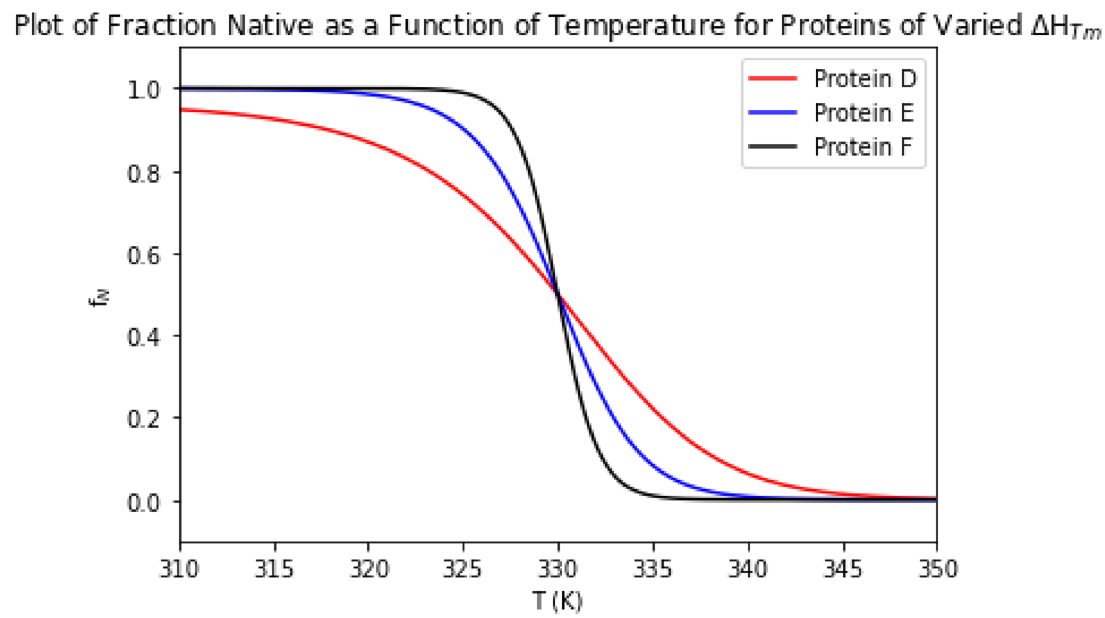


Part B

> Protein F has the steepest transition because the slope of the transition is set by ΔH_{Tm} . According to equation 8.47 in the textbook, the first temperature derivative of the fraction native function is $\frac{K}{(1+K)^2}\ \frac{\Delta\overline{H}}{RT^2}$ and thus as the magnitude of ΔH_{Tm} increases, so does the steepness of the unfolding curve.

[9]:
```
def fraction_native(T, Tm, DHtm, DCp):
    R = 1.987204e-3 #IDG constant in kcal/mol K
    DGb = DHtm + (DCp*(T-Tm)) - ((T*DHtm)/Tm) - (T*DCp*np.log(T/Tm))
    K = np.e ** ((-1*DGb)/(R*T))
    fn = K / (K+1)
    return fn


Ts = np.arange(300,500,0.01)
D = fraction_native(Ts, 330, -50, -2)
E = fraction_native(Ts, 330, -100, -2)
F = fraction_native(Ts, 330, -200, -2)

plt.plot(Ts, D, 'r', label="Protein D")
plt.plot(Ts, E, 'b', label="Protein E")
plt.plot(Ts, F, 'k', label="Protein F")
plt.axis([310,350,-0.1,1.1])
plt.ylabel("f$_{N}$")
plt.xlabel("T (K)")
```

```
plt.title("Plot of Fraction Native as a Function of Temperature for Proteins of␣
↪Varied $\Delta$H$_{Tm}$")
plt.legend();
```

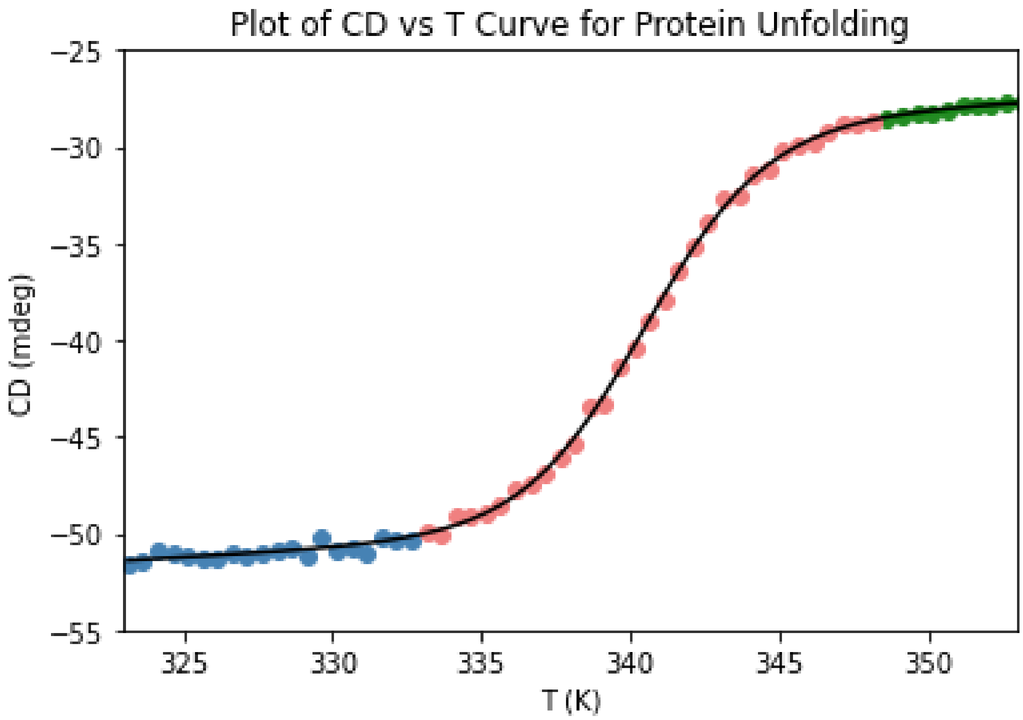


0.0.4 Problem 5.5

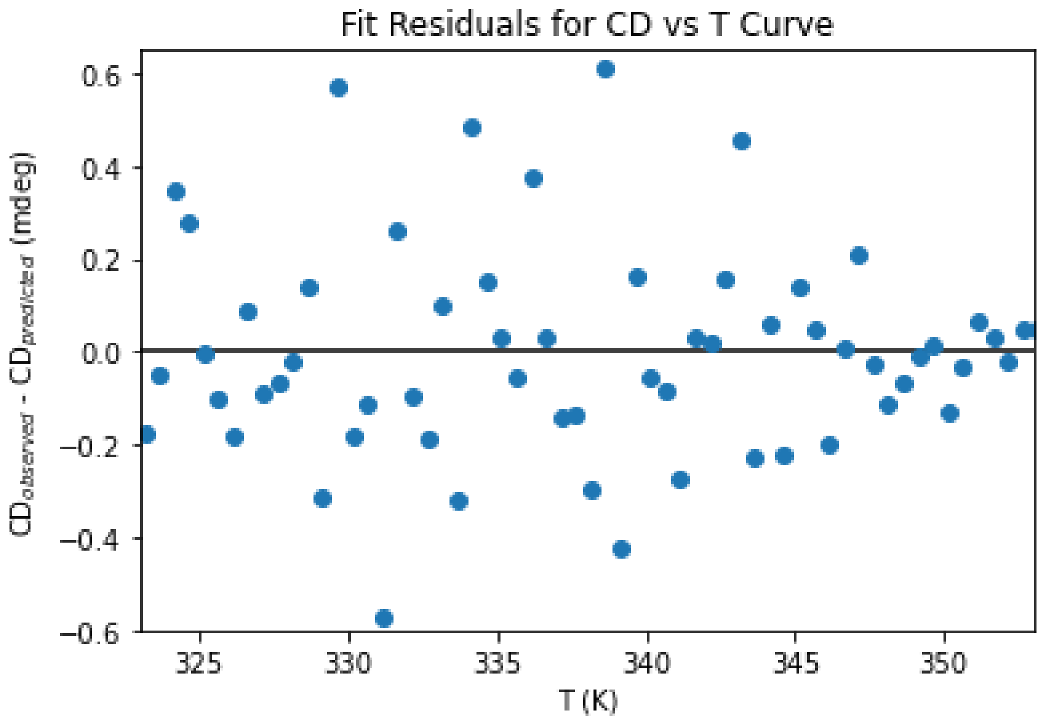
Part A

[10]:
```
lyz_data = pd.read_csv('./lyz_pH25.txt', sep='\t', header=None)
X = np.array(lyz_data[0])

X = np.array([i + 273.15 for i in X])
Y = np.array(lyz_data[1])

plt.plot(X[20:-10], Y[20:-10], 'o', color='lightcoral')
plt.ylabel("CD (mdeg)")
plt.xlabel("T (K)")
plt.title("Plot of CD vs T Curve for Protein Unfolding")

def Yn(T, a, b):
    Yn = a*T + b
    return Yn

def Yd(T, c, d):
    Yd = c*T + d
    return Yd
```

```python
def lysozymefit(T, Tm, DH, a, b, c, d):
    YN = a*T + b
    YD = c*T + d
    R = 8.3145e-3 #KJ/mol K
    K = np.exp((DH * (T-Tm))/(R*T*Tm))
    #print(K)
    Yobs = YD + (((YN - YD)*K)/(1+K))
    return Yobs

#N:[:20] D:[-10:]
Nx = np.array(X[:20])
Ny = np.array(Y[:20])
Nguess = [1, -60]
npopt, npcov = curve_fit(Yn, Nx, Ny, Nguess)
#plt.plot(Nx, Yn(Nx, *npopt), 'steelblue')
plt.scatter(Nx, Ny, color='steelblue')

Dx = np.array(X[-10:])
Dy = np.array(Y[-10:])
Dguess = [2, -40]
dpopt, dpcov = curve_fit(Yd, Dx, Dy, Dguess)
plt.plot(Dx, Yd(Dx, *dpopt), 'g')
plt.scatter(Dx, Dy, color='forestgreen')
#print(npopt, dpopt)

guesses = [342, -400, npopt[0], npopt[1], dpopt[0], dpopt[1]]

popt, pcov = curve_fit(lysozymefit, X, Y, guesses)
#popt[1] = -2179 #sets the DH to the ideal from Cp given in  5.5B. Is very␣
↪clearly not correct for this data.
plt.plot(X, lysozymefit(X, *popt) ,'k');
plt.axis([323,353,-55, -25])

'''perr = np.sqrt(np.diag(pcov))
for i,j in enumerate(popt):
    print(f'{j} +/- {perr[i]}')''';
```

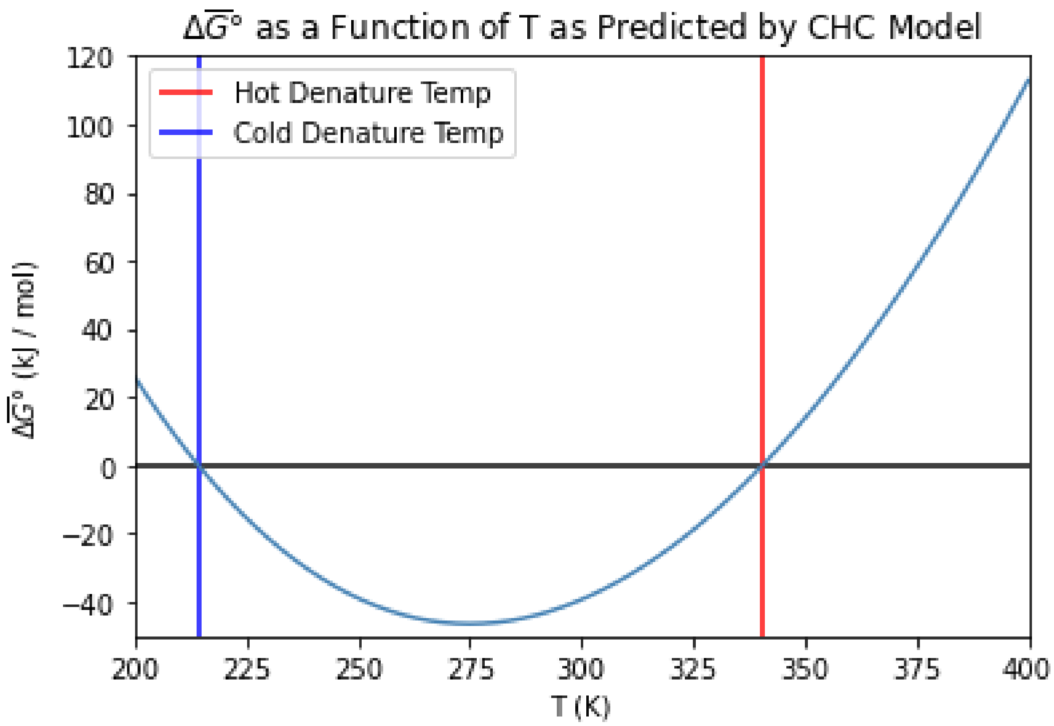


[11]:
```
residuals = Y - lysozymefit(X, *popt)
plt.plot(X, residuals, 'o')
plt.hlines(0,300,400, 'k')
plt.axis([323,353,-0.6, 0.65])
plt.title("Fit Residuals for CD vs T Curve")
plt.xlabel("T (K)")
plt.ylabel("CD$_{observed}$ - CD$_{predicted}$ (mdeg)");
```



Part B

[12]:
```
DH_pred = popt[1]
Tm_pred = popt[0]
DCp = -6.4 #kJ/mol K

def DGstdbar(T, Tm, DH, Cp):
    return DH + (Cp*(T-Tm)) - (T*DH/Tm) - T*Cp*np.log(T/Tm)

Ts = np.arange(200, 401, 1)
Gs = DGstdbar(Ts, Tm_pred, DH_pred, DCp)

plt.plot(Ts, Gs, 'steelblue')
plt.title("$\Delta$$\overline{G}$$\degree$ as a Function of T as Predicted by␣
↪CHC Model")
plt.xlabel("T (K)")
plt.ylabel("$\Delta$$\overline{G}$$\degree$ (kJ / mol)")
plt.axis([200, 400, -50, 120]);
plt.hlines(0,100,500,'k');
plt.vlines(Tm_pred, -50, 130, color='r', label="Hot Denature Temp")
plt.vlines(214.326, -50, 130, color='b', label="Cold Denature Temp")
plt.legend();
```



Part C

[17]:
```
DG_350 = DGstdbar(350, Tm_pred, DH_pred, DCp)
DG_310 = DGstdbar(310, Tm_pred, DH_pred, DCp)

print(DG_350, 'kJ/mol', DG_310, 'kJ/mol')
```

```
13.78510531219272 kJ/mol -32.72397268695846 kJ/mol
```

Part D

[14]:
```
DGstdbar(214.32587, Tm_pred, DH_pred, DCp)
#should be zero and it is
```

[14]: 7.80285347445897e-06