TiO Properties

October 13, 2020

1 MATSE 580 Homework 1

1.1 Jonathan Boualavong

Thermodynamics of a binary system: $Ti - O$

Data from: A. Watson and T. Markus, Eds., "Ternary System Fe-Mo-Ti: Datasheet from Landolt-BÃűrnstein - Group IV Physical Chemistry ÂůVolume 19C2: "Ternary Steel Systems: Phase Diagrams and Phase Transition Data'' in SpringerMaterials (https://doi.org/10.1007/978-3-540-88154-4_20)." Springer-Verlag Berlin Heidelberg, doi: 10.1007/978-3-540-88154-4_20.

Assignment: a). Calculate the heat capacity, entropy, enthalpy and Gibbs energy of each pure element as a function of temperature. b). Calculate the phase fractions as a function of temperature for the alloy with 50/50 of each element. c). Calculate the enthalpy of formation as a function of composition at 298K. d). Calculate the enthalpy of mixing in the liquid phase at a high temperature of your choice. e). Calculate the binary phase diagram and label all phase regions of one, two, and three phases in equilibrium.

[1]:
```
# Set up the database and primary variables

# Load in the database file
from pycalphad import Database
db = Database("TiO_Complete.tdb")
print(sorted(db.elements))
print(sorted(db.phases))

# Composition: copper, oxygen, vacancies
#comp = sorted(db.elements)
# Relevant phases
#phases = sorted(db.phases)
```

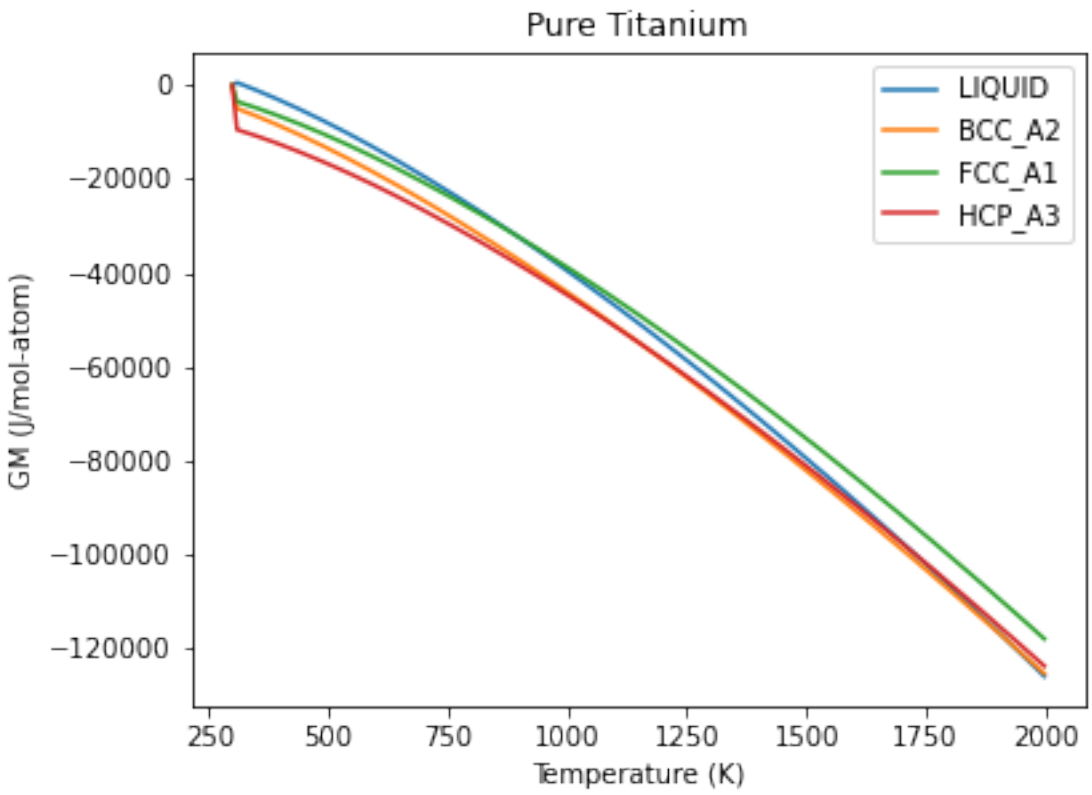
```
['/-', 'O', 'TI', 'VA']
['A_TIO', 'BCC_A2', 'FCC_A1', 'GAS', 'HCP_A3', 'I_LIQUID', 'LIQUID', 'TI2O3',
'TI3O2', 'TI3O5', 'TIO2', 'TIOX']
```

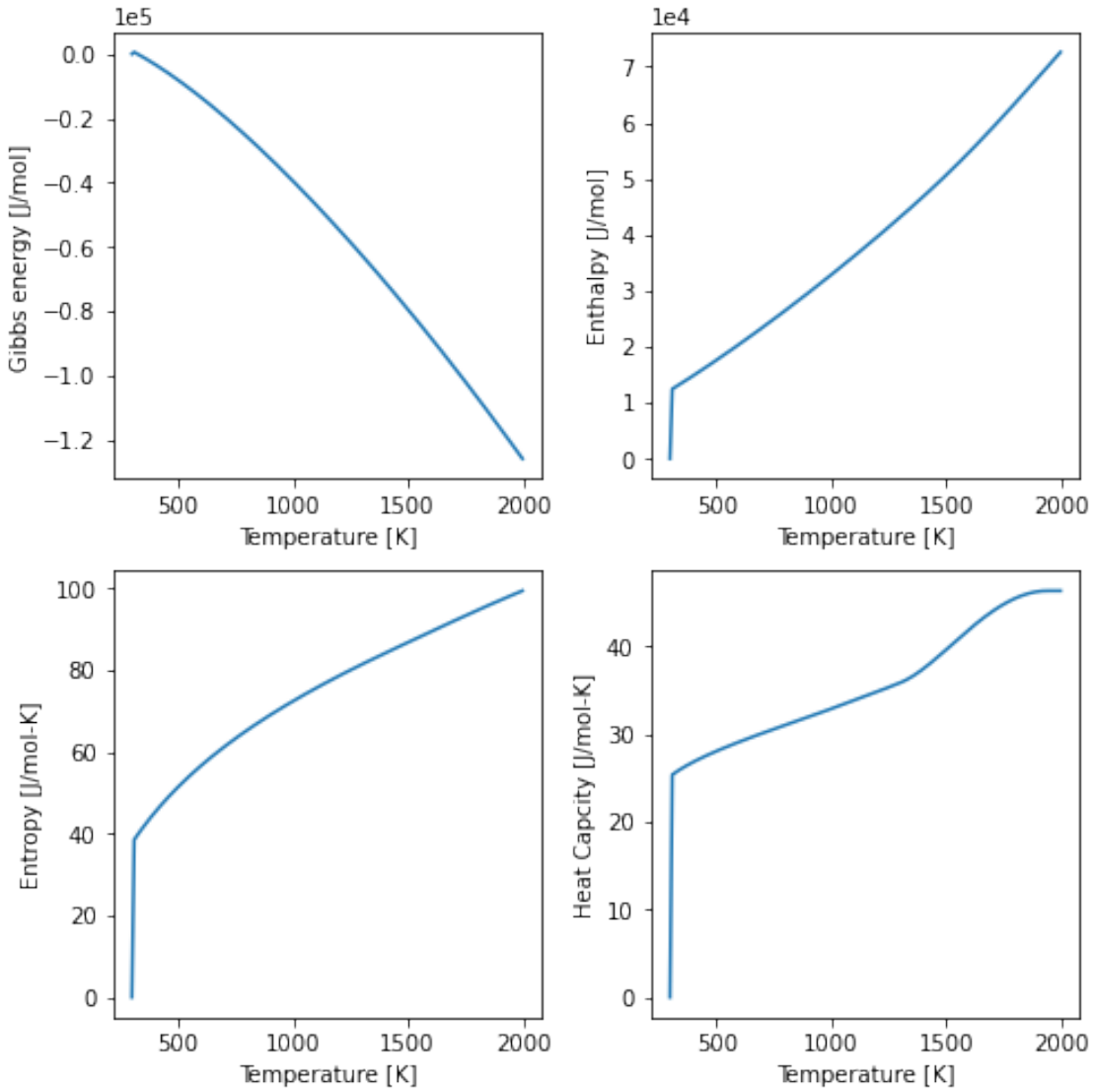
1.1.1 Part a) Pure element properties

Heat capacity, entropy, enthalpy, and Gibbs energy as a function of temperature.

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
from pycalphad import Database, calculate, equilibrium, variables as v
#from PT import PT_phase_diagram

# Conditions
P = 101325 # [=] Pa, pressure = 1 atm
T_lo = 298 # [=] K, low temperature bound
T_hi = 2000 # [=] K, high temperature bound
N = 1 # [=] mole

## Properties of pure titanium
# Relevant phases for pure Ti:
phases_Ti = ['LIQUID', 'BCC_A2', 'FCC_A1', 'HCP_A3']

# Gibbs energy of different phases
plt.figure(figsize = [6, 4.5])
for phase in phases_Ti:
    Ti_GM = calculate(db, ['TI', 'VA'], phase, P=P, T=(T_lo, T_hi, 10), N=N)
    plt.plot(Ti_GM.T, Ti_GM.GM.squeeze(), label = phase)
plt.legend()
plt.xlabel('Temperature (K)')
plt.ylabel('GM (J/mol-atom)')
plt.title('Pure Titanium')
plt.show()

# Equilibrium characteristics
conditions = {  # mapping of state variable to values
    v.N: 1,
    v.P: 101325,
    # for now, only temperature has a range. If multiple variables have ranges,␣
 ↪all combinations are calculated
    v.T: (T_lo, T_hi, 10),
}
# Start figure
fig, ax = plt.subplots(figsize = [7, 7], sharex=True)

# Gibbs energy
Ti_gm = equilibrium(db, ['Ti'], phases_Ti, conditions, output='GM')
plt.subplot(2,2,1)
plt.plot(Ti_gm.T, Ti_gm.GM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Gibbs energy [J/mol]")
```

```
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Heat capacity
Ti_cp = equilibrium(db, ['Ti'], phases_Ti, conditions, output='CPM')
plt.subplot(2,2,4)
plt.plot(Ti_cp.T, Ti_cp.CPM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Heat Capcity [J/mol-K]")
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Entropy
Ti_sm = equilibrium(db, ['Ti'], phases_Ti, conditions, output='SM')
plt.subplot(2,2,3)
plt.plot(Ti_sm.T, Ti_sm.SM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Entropy [J/mol-K]")
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Enthalpy
Ti_hm = equilibrium(db, ['Ti'], phases_Ti, conditions, output='HM')
plt.subplot(2,2,2)
plt.plot(Ti_hm.T, Ti_hm.HM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Enthalpy [J/mol]")
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Space the figure out more for ease of viewing
fig.tight_layout(pad=1.0)
plt.show()
```

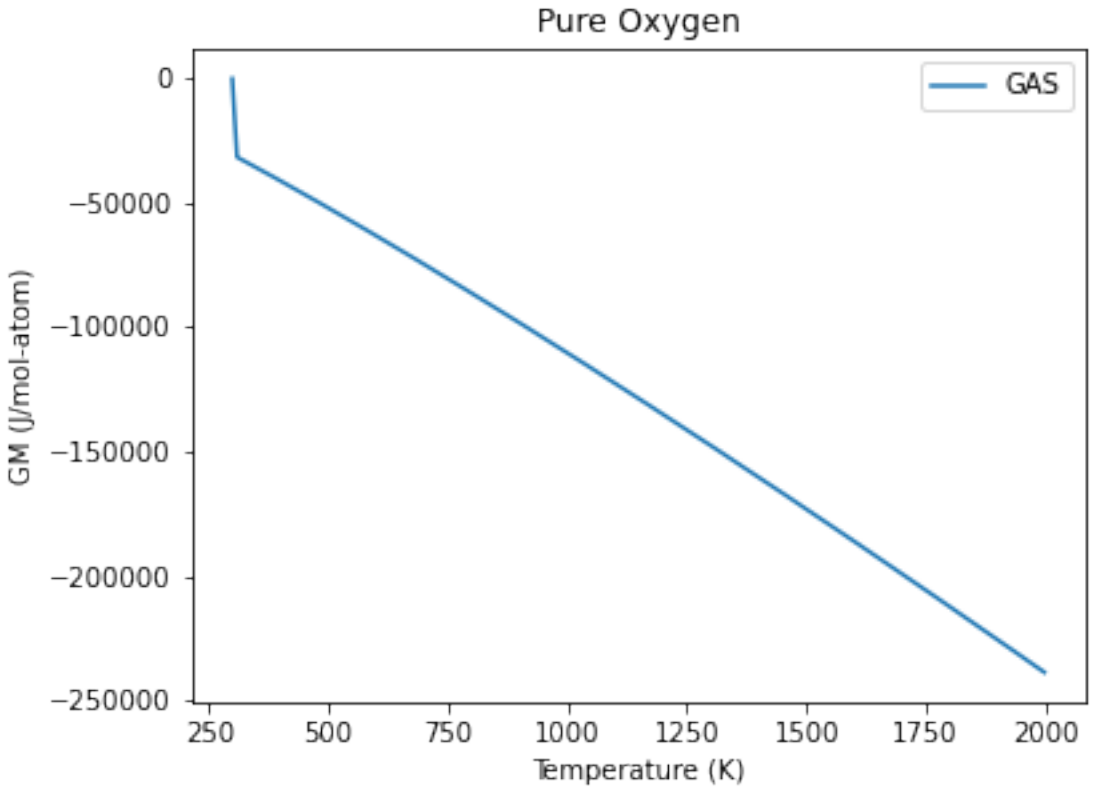

Pure Titanium
LIQUID
BCC_A2
FCC_A1
HCP_A3
GM (J/mol-atom)
0
−20000
−40000
−60000
−80000
−100000
−120000
250
500
750
1000
1250
1500
1750
2000
Temperature (K)

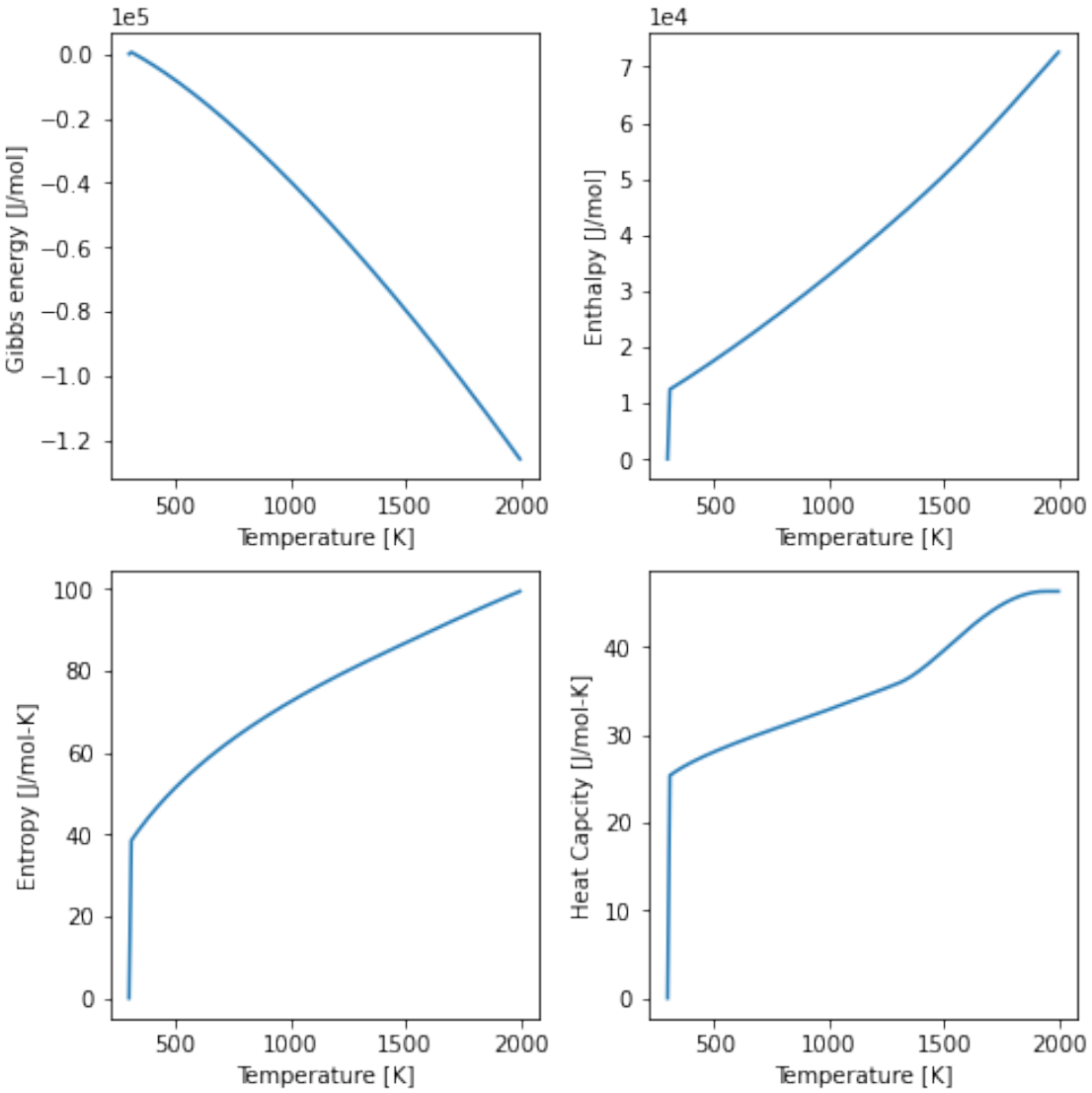


[3]:
```
# Repeat for oxygen
import matplotlib.pyplot as plt
import numpy as np
from pycalphad import Database, calculate, equilibrium, variables as v
#from PT import PT_phase_diagram

# Conditions
P = 101325 # [=] Pa, pressure = 1 atm
T_lo = 298 # [=] K, low temperature bound
T_hi = 2000 # [=] K, high temperature bound
N = 1 # [=] mole

## Properties of pure oxygen
```

```
# Relevant phases for pure O:
phases_O = ['GAS']

# Gibbs energy of different phases
plt.figure(figsize = [6, 4.5])
for phase in phases_O:
    O_GM = calculate(db, ['O', 'VA'], phase, P=P, T=(T_lo, T_hi, 10), N=N)
    plt.plot(O_GM.T, O_GM.GM.squeeze(), label = phase)
plt.legend()
plt.xlabel('Temperature (K)')
plt.ylabel('GM (J/mol-atom)')
plt.title('Pure Oxygen')
plt.show()

# Equilibrium characteristics
conditions = {  # mapping of state variable to values
    v.N: 1,
    v.P: 101325,
    # for now, only temperature has a range. If multiple variables have ranges,␣
 ↪all combinations are calculated
    v.T: (T_lo, T_hi, 10),
}
# Start figure
fig, ax = plt.subplots(figsize = [7, 7], sharex=True)

# Gibbs energy
O_gm = equilibrium(db, ['O'], phases_O, conditions, output='GM')
plt.subplot(2,2,1)
plt.plot(Ti_gm.T, Ti_gm.GM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Gibbs energy [J/mol]")
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Heat capacity
O_cp = equilibrium(db, ['O'], phases_O, conditions, output='CPM')
plt.subplot(2,2,4)
plt.plot(Ti_cp.T, Ti_cp.CPM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Heat Capcity [J/mol-K]")
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Entropy
O_sm = equilibrium(db, ['O'], phases_O, conditions, output='SM')
plt.subplot(2,2,3)
plt.plot(Ti_sm.T, Ti_sm.SM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Entropy [J/mol-K]")
```

```
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Enthalpy
O_hm = equilibrium(db, ['O'], phases_O, conditions, output='HM')
plt.subplot(2,2,2)
plt.plot(Ti_hm.T, Ti_hm.HM.squeeze())
plt.xlabel("Temperature [K]")
plt.ylabel("Enthalpy [J/mol]")
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])

# Space the figure out more for ease of viewing
fig.tight_layout(pad=1.0)
plt.show()
```



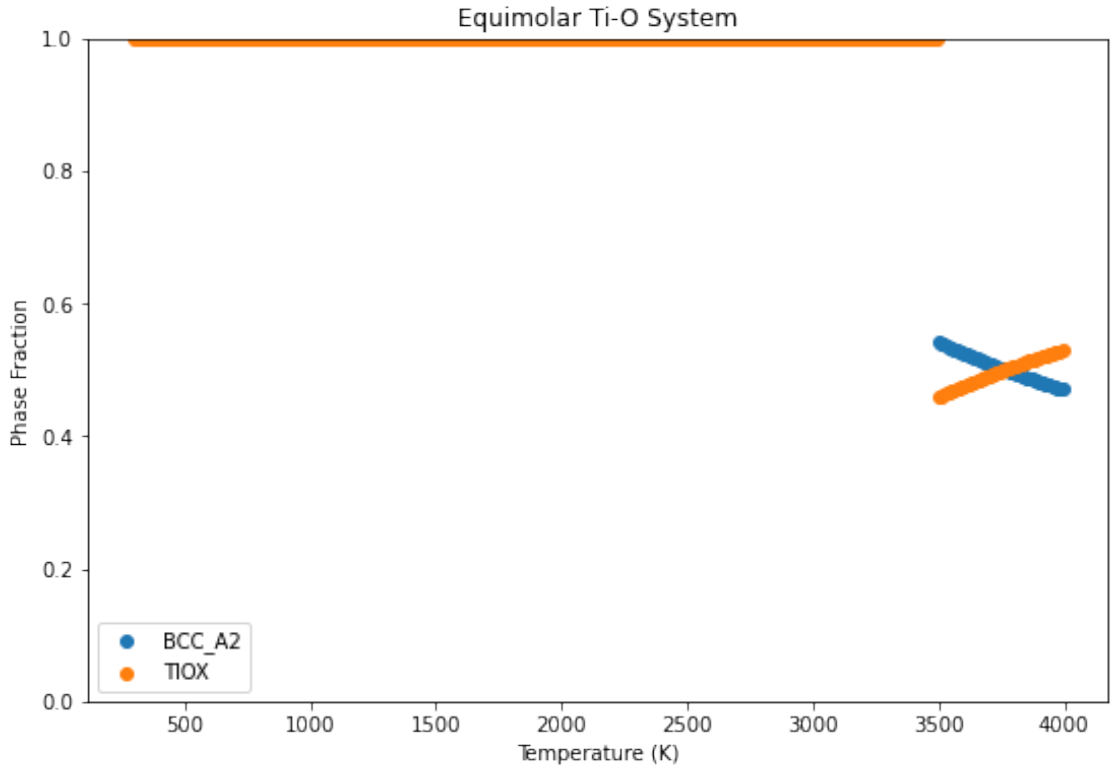


1.1.2 Part b) Equimolar mixture phase fractions

as a function of temperature

[12]:
```
import matplotlib.pyplot as plt
import numpy as np
from pycalphad import Database, calculate, equilibrium, variables as v

# Equilibrium characteristics
conditions = {  # mapping of state variable to values
    v.N: 1,
    v.P: 101325,
```

```
    v.T: (300, 4000, 10), # Database is limited to about 4000 K for its equations
    v.X('O'): 0.5
}
phases_mix = ['A_TIO', 'BCC_A2', 'FCC_A1', 'GAS', 'HCP_A3', 'I_LIQUID',␣
 ↪'LIQUID', 'TI2O3', 'TI3O2', 'TI3O5', 'TIO2', 'TIOX']
phase_eq = equilibrium(db, ['TI', 'O', 'VA'], phases_mix, conditions)

fig = plt.figure(figsize=(9,6))
ax = fig.gca()
for phase in phases_mix:
    # Phase indices for plotting
    phase_indices = np.nonzero(phase_eq.Phase.values == phase)
    # Many phases are not present in any appreciable amount; omit if empty so␣
 ↪the legend is not too large
    if len(phase_eq.NP.values[phase_indices]) > 0:
        plt.scatter(np.take(phase_eq['T'].values, phase_indices[2]), phase_eq.NP.
 ↪values[phase_indices], label = phase)
plt.legend(loc = 'lower left')
plt.title('Equimolar Ti-O System')
plt.ylabel('Phase Fraction')
plt.ylim(0, 1)
plt.xlabel('Temperature (K)')
plt.show()
```

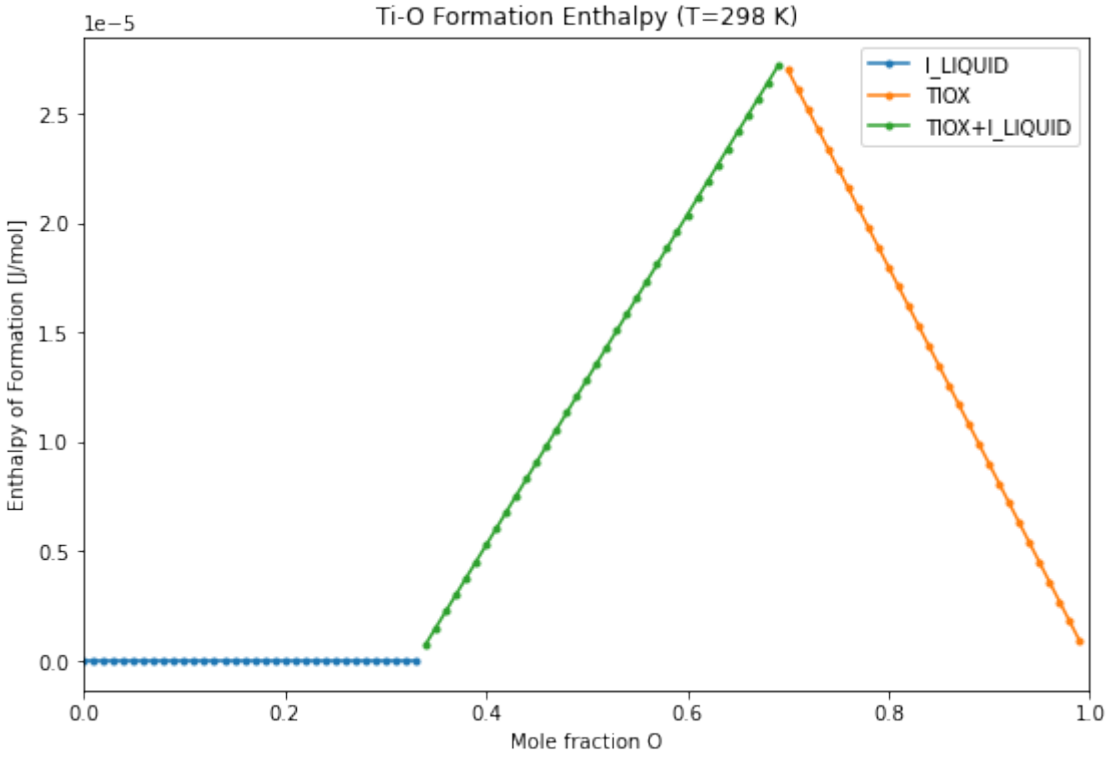


1.1.3 Part c) Enthalpy of formation

T = 298 K as a function of composition

[6]:
```
from pycalphad import Database, equilibrium, Model, ReferenceState, variables as v
import matplotlib.pyplot as plt
import numpy as np

db = Database("TiO_Complete.tdb")
comps = ["TI", "O", "VA"]
phases = db.phases.keys()

# Single element reference state phase for Ti and O
Ti_ref = ReferenceState("TI", "HCP_A3", {v.T: 298.15, v.P: 101325})
O_ref  = ReferenceState("O",  "GAS", {v.T: 298.15, v.P: 101325})

# Create the models for each phase and shift them all by the same reference states.
eq_models = {}
for phase_name in phases:
    mod = Model(db, comps, phase_name)
    mod.shift_reference_state([Ti_ref, O_ref], db)
    eq_models[phase_name] = mod

# Calculate HMR at 298 K from X(O)=0 to X(O)=1
conds = {v.P: 101325, v.T: 298, v.X("O"): (0, 1, 0.01)}
result = equilibrium(db, comps, phases, conds, output="HMR", model=eq_models)

# Find the groups of unique phases in equilibrium
unique_phase_sets = np.unique(result.Phase.values.squeeze(), axis=0)

# Plot
fig = plt.figure(figsize=(9,6))
ax = fig.gca()
for phase_set in unique_phase_sets:
    label = '+'.join([ph for ph in phase_set if ph != ''])
    # Composition indices with the same unique phase
    unique_phase_idx = np.nonzero(np.all(result.Phase.values.squeeze() == phase_set, axis=1))[0]
    masked_result = result.isel(X_O=unique_phase_idx)
    ax.plot(masked_result.X_O.squeeze(), masked_result.HMR.squeeze(), marker='.', label=label)
ax.set_xlim((0, 1))
```

```
ax.set_xlabel('Mole fraction O')
ax.set_ylabel('Enthalpy of Formation [J/mol]')
ax.set_title('Ti-O Formation Enthalpy (T=298 K)')
plt.ticklabel_format(axis = 'y', style = 'sci', scilimits = [-3, 3])
ax.legend()
plt.show()
```

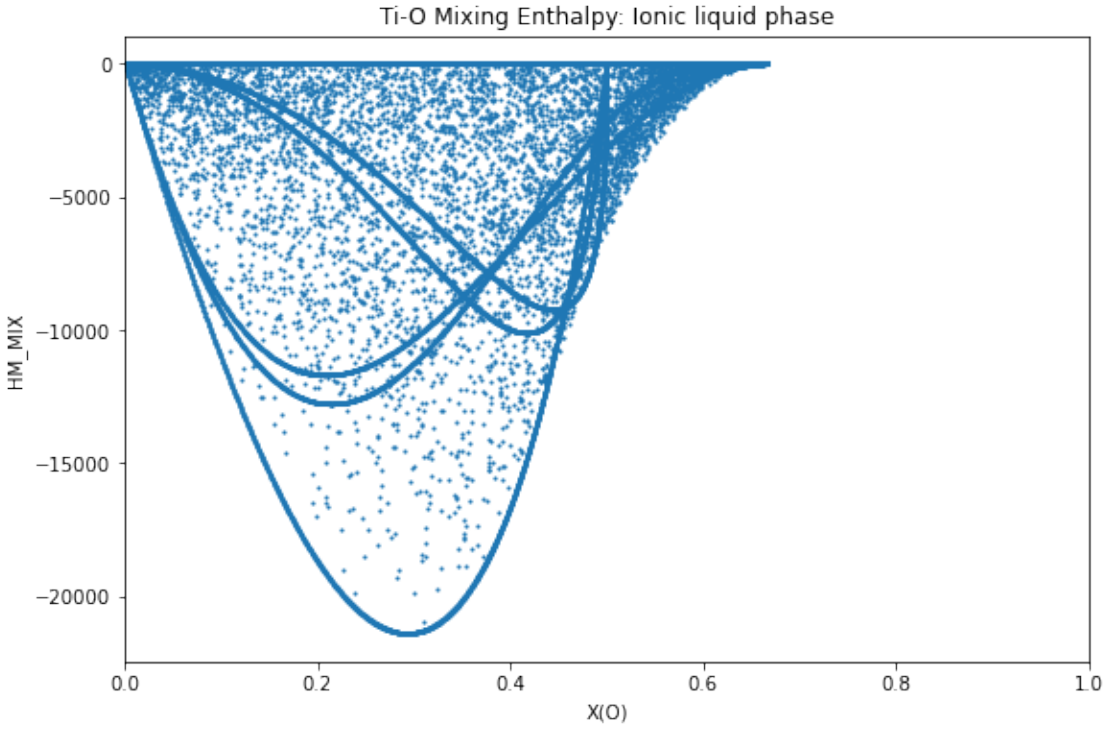


1.1.4 Part d) Enthalpy of mixing, liquid phase

Based on the phase fractions above, use T = 6500 K

[20]:
```
from pycalphad import Database, calculate
import matplotlib.pyplot as plt

dbf = Database("TiO_Complete.tdb")
comps = ["TI", "O", "VA"]

# Calculate HMR for the Chi at 2800 K from X(RE)=0 to X(RE)=1
result = calculate(dbf, comps, "I_LIQUID", P=101325, T=2500, output='HM_MIX')

# Plot
```

```
fig = plt.figure(figsize=(9,6))
ax = fig.gca()
ax.scatter(result.X.sel(component='O'), result.HM_MIX, marker='.', s=5,␣
 ↪label='I_LIQUID')
ax.set_xlim((0, 1))
ax.set_xlabel('X(O)')
ax.set_ylabel('HM_MIX')
ax.set_title('Ti-O Mixing Enthalpy: Ionic liquid phase')
plt.show()
```

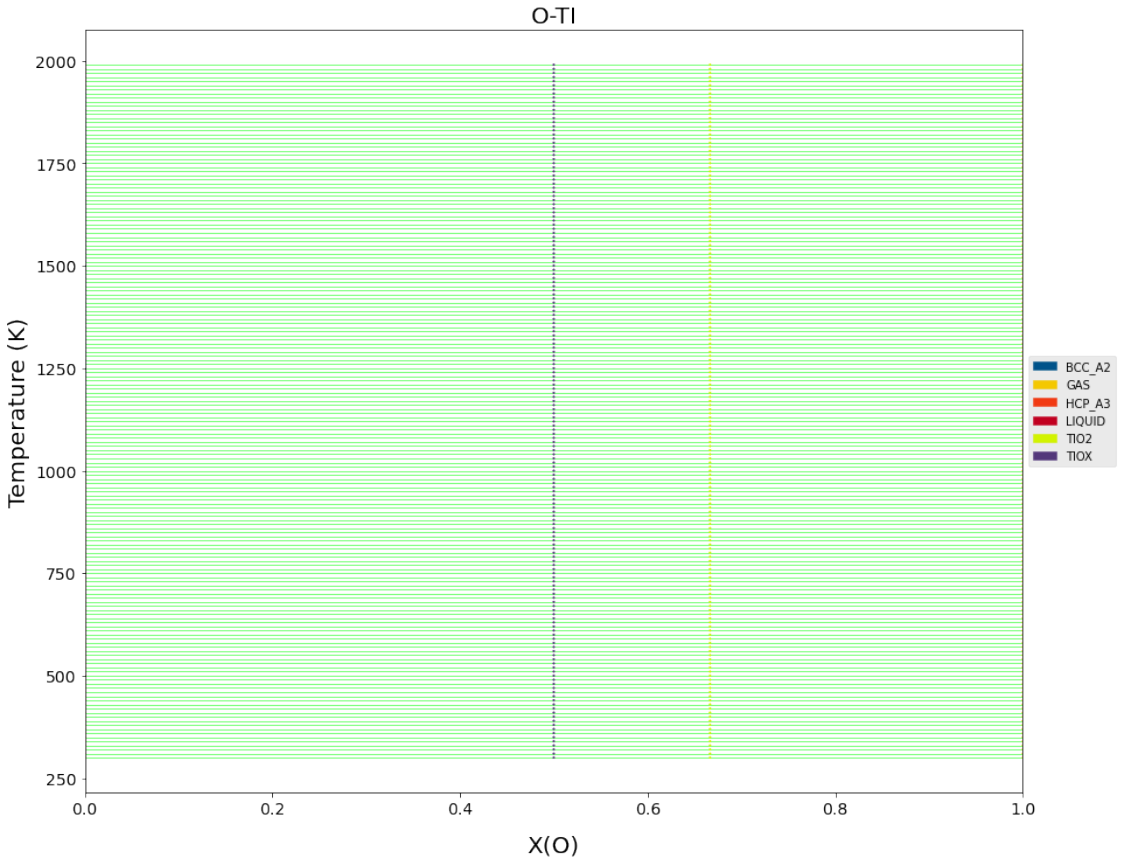


1.1.5 Part e) Binary phase diagram

Label all phase regions of 1, 2, and 3, phase equilibrium

[22]:
```
import matplotlib.pyplot as plt
from pycalphad import Database, binplot
import pycalphad.variables as v

# Load database and choose the phases that will be considered
db = Database('TiO_Complete.tdb')
phases_rel = ['BCC_A2', 'GAS', 'HCP_A3', 'LIQUID', 'TIO2', 'TIOX']
```

```
# Create a matplotlib Figure object and get the active Axes
fig = plt.figure(figsize=(18,12))
axes = fig.gca()

# Compute the phase diagram and plot it on the existing axes using the␣
 ↪`plot_kwargs={'ax': axes}` keyword argument
plt.style.use('ggplot')
binplot(db, ['TI', 'O', 'VA'] , phases_rel, {v.X('O'):(0,1,0.02), v.T: (300,␣
 ↪2000, 10), v.P:101325, v.N: 1}, plot_kwargs={'ax': axes})

plt.show()
```



[ ]: