# Informações para o TCC - Isabela

# Modelos termodinâmicos de acordo com o modelo CALPHAD

Neste trabalho usaremos modelos para fases do tipo solução e fases estequiométricas, descritos a seguir.

## Fases do tipo solução

A energia molar de Gibbs de uma fase to tipo solução é   
Gmϕ = Grefϕ + Gidϕ + Gexϕ  
sendo Grefϕ a energia molar de Gibbs de referência:   
Grefϕ = xMoϕ°GMoϕ + xZrϕ°GZrϕ  
onde xiϕ é a fração molar do elemento i na fase ϕ e °Giϕ a energia molar de Gibbs do elemento i puro em relação ao estado de referência escolhido. Gidϕ a energia molar de Gibbs ideal:   
Gidϕ = RT(xMoϕlnxMoϕ+xZrϕlnxZrϕ)  
sendo R = 8, 31451 J/mol/K e T a temperatura absoluta. Finalmente, Gexϕ a energia molar de Gibbs de excesso:   
Gexϕ = xMoϕxZrϕ[L0ϕ+L1ϕ(xMoϕ−xZrϕ)+L2ϕ(xMoϕ−xZrϕ)2+…]  
onde os parâmetros Liϕ são geralmente funções lineares da temperatura na forma   
Liϕ = Aiϕ + BiϕT  
sendo que frequentemente adota-se Biϕ = 0.

## Fases estequiométricas

O modelo para fases estequiométricas é parecido com o modelo de fases do tipo solução, mas o termo ideal se anula.

# Descrição termodinâmica do sistema Mo-Zr

Segundo (KAUFMAN e BERNSTEIN, 1970), o sistema binário Mo-Zr é descrito usando o conjunto de equações abaixo para as fases de equiĺíbrio.

## Fases do tipo solução

Neste caso, o estado padrão de referência para os elementos puros é sempre a fase líquida e todas as fases são descritas como soluções regulares, ou seja, apenas os termos L0ϕ estão presentes.

### Líquido

Como o líquido é a fase de referência para os elementos puros, temos que   
Grefliq = 0  
e o termo de excesso é   
Gexliq = 1512xMoliqxZrliq

### Fase CCC

°GMoCCC =  − 5800 + 2T

°GZrCCC =  − 4250 + 2T

GexCCC = 6551xMoCCCxZrCCC

### Fase HCP

°GMoHCP =  − 3800 + 2T

°GZrHCP =  − 5280 + 2.9T

GexHCP = 8981xMoHCPxZrHCP

## Fases estequiométricas

### Fase de Laves

Apenas a fase de Laves Mo2Zr é considerada estequiométrica no sistema. A referência para os elementos puros é neste caso a fase HCP, de forma que

°GMoLaves =  − 3800 + 2T

°GZrLaves =  − 5280 + 2.9T

GexLaves =  − 16488xMoLavesxZrLaves

# Anexo: códigos

## Algoritmos para a determinação do envoltório convexo e das tielines

inserir comentários sobre o código

import numpy as np

import scipy.spatial as spa

def hull(x, T, data):

# getting minimal G for each x - solution phases

Fgy = [fase.G(x, T) for fase in data if fase.kind == 'sol']

Gmin = np.amin(Fgy, axis=0)

Gmin = np.column\_stack((x, Gmin))

# adding stoichiometric phases

for fase in data:

if fase.kind == 'stq':

point = [[fase.xB, fase.G(T)]]

Gmin = np.append(Gmin, point, axis=0)

# ordering array by composition

Gmin = Gmin[np.argsort(Gmin[:, 0])]

# getting convex hull for solution phases

hull = spa.ConvexHull(Gmin)

# shifting first position to end

vertices = np.roll(hull.vertices, -1)

# separating points belonging to hull

points = hull.points[vertices, :]

return Gmin, points, vertices

def tielines(vertices, Gdata, dx, TOL=1e-8):

# getting differences in vertices array

t = np.diff(vertices)

t = np.where(t > 1)[0] # filtering only diffs > 1

tielines = [] # initializing tielines list

for i in t:

# checking for tielines based on dx + TOL

if Gdata[vertices[i + 1], 0] - Gdata[vertices[i], 0] > dx + TOL:

tielines.append(vertices[i])

tielines.append(vertices[i + 1])

nt = len(tielines)

return tielines, nt

## Definições de modelos de fases segundo o método CALPHAD

inserir comentários sobre o código

import numpy as np

'''

CALPHAD phase models

'''

class sol\_phase:

'''

Solution phases

'''

def \_\_init\_\_(self, label, Gref, L, R=1.987):

self.label = label

self.L = np.array(L)

self.Gref = np.array(Gref)

self.R = R

self.kind = 'sol'

def xlnx(self, x):

s = x \* np.log(x)

return np.nan\_to\_num(s)

def Sid(self, x):

xA = 1 - x

xB = x

f = self.xlnx(xA) + self.xlnx(xB)

return -self.R \* f

def Gex(self, x, T):

xA = 1 - x

xB = x

c = xA - xB

cp = 1

p = 0

for n in range(self.L.size):

Ln = eval(self.L[n])

p += Ln \* cp

cp \*= c

return p \* xA \* xB

def G(self, x, T):

Gex = self.Gex(x, T)

GA = eval(self.Gref[0])

GB = eval(self.Gref[1])

Gref = (1 - x) \* GA + x \* GB

return Gref + Gex - T \* self.Sid(x)

class stq\_phase:

'''

Stoichiometric phases

'''

def \_\_init\_\_(self, label, n, Gref, L):

self.label = label

self.xA = n[0] / (n[0] + n[1])

self.xB = 1 - self.xA

self.Gref = np.array(Gref)

self.L = np.array(L)

self.kind = 'stq'

def Gex(self, T):

c = self.xA - self.xB

cp = 1

p = 0

for n in range(self.L.size):

Ln = eval(self.L[n])

p += Ln \* cp

cp \*= c

return p \* self.xA \* self.xB

def G(self, T):

GA = eval(self.Gref[0])

GB = eval(self.Gref[1])

Gref = self.xA \* GA + self.xB \* GB

return Gref + self.Gex(T)

## Código para gerar curvas de energia livre molar vs. composição

inserir comentários sobre o código

import numpy as np

import convexhull as ch

import matplotlib.pyplot as plt

import markers

import MoZr

# importing system data

MoZr = MoZr.data()

# Calculation of free energy curves

Tmin, Tmax, dT = 1200, 3000, 50 # temperature limits for the phase diagram

Nx, tol = 50, 1e-6 # number of compositions, first composition

x = np.linspace(tol, 1-tol, Nx+1) # composition array

dx = x[1] - x[0]

# initializing graphics

fig = plt.figure()

ax = fig.add\_subplot(111)

# creating G curves

Tlist = np.arange(Tmin, Tmax+dT, dT)

for T in Tlist:

print(f'T = {T} K')

# figure name

figfile = f'../example/PhD/MoZr-T{T}.png'

# preparing graphics

plt.cla()

ax.set\_xlim([0, 1])

ax.set\_xlabel(r'$x\_\mathrm{Zr}$')

ax.set\_ylabel(r'$G\_m$ (J/mol)')

ax.text(0.03, .95, f'$T={T}$ K', transform=ax.transAxes)

# calculating hull and tielines

Gdata, hull, vertices = ch.hull(x, T, MoZr)

tielines, nt = ch.tielines(vertices, Gdata, dx)

# showing free energy curves

mkr = markers.markers()

for phase in MoZr:

if phase.kind == 'sol':

ax.plot(x, phase.G(x, T), next(mkr), markersize=4,

label=phase.label)

elif phase.kind == 'stq':

ax.plot(phase.xB, phase.G(T), 'o', label=phase.label, markersize=8)

# showing tielines

for i in range(0, nt, 2):

ax.plot(Gdata[tielines[i:i+2], 0], Gdata[tielines[i:i+2], 1],

'b-o', markersize=6)

ax.plot([], [], 'b-o', label='tielines') # legend

# showing convex hull

ax.plot(hull[:, 0], hull[:, 1], 'k.', label='convex hull')

plt.legend()

fig.tight\_layout()

plt.savefig(figfile)

## Código para calcular um diagrama de fases binário

inserir comentários sobre o código

import numpy as np

import convexhull as ch

import matplotlib.pyplot as plt

import MoZr

MoZr = MoZr.data() # loading system data

Tmin, Tmax = 500, 3000 # temperature limits for the phase diagram

# initializing graphics

fig = plt.figure()

ax = fig.add\_subplot(111)

# lists of precision parametes to consider

dTlist = [20, 10, 5, 1, .5]

Nxlist = [50, 100, 500, 1000, 5000, 10000]

# starting to generate figures

for dT in dTlist:

for Nx in Nxlist:

# starting graph

plt.cla()

ax.set\_xlim([0, 1])

ax.set\_ylim([Tmin, Tmax])

ax.set\_xlabel(r'$x\_\mathrm{Zr}$')

ax.set\_ylabel(r'$T$ (K)')

# figure name

figfile = f'../example/Gcurves/MoZr-dT{dT}-Nx{Nx:05d}.png'

print(figfile)

Temps = np.arange(Tmin, Tmax + dT, dT) # temperature array

tol = 1e-6 # first composition (to avoid log(0) calculation)

x = np.linspace(tol, 1 - tol, Nx + 1) # composition array

dx = x[1] - x[0]

# starting calculations

for T in Temps:

if T % 250 == 0: # progress indicator

print(f'T = {T} K')

# calculating hull and tielines

Gdata, hull, vertices = ch.hull(x, T, MoZr)

tielines, nt = ch.tielines(vertices, Gdata, dx)

# adding points to phase diagram

ax.plot(Gdata[tielines, 0], [T] \* nt, 'k.', markersize=1)

fig.tight\_layout()

plt.savefig(figfile)

## Código auxiliar para a geração de marcadores nos gráficos

inserir comentários sobre o código

def markers():

'''

Function to generate markers sequentially

'''

mkr\_list = ['o', 's', 'v', '^', '<', '>', 'P', '\*', 'D']

N = len(mkr\_list)

n = 0

while n < N:

yield mkr\_list[n] + '-'

n += 1

else:

n = 0

yield mkr\_list[n] + '-'

## Exemplo de um sistema: Mo-Zr (KAUFMAN e BERNSTEIN, 1970)

inserir comentários sobre o código

import thermo

'''

Data from

Kaufman, L., & Bernstein, H. (1970). Computer calculation of phase

diagrams, with special reference to refractory metals.

Academic Press, New York.

'''

def data():

GMoLiq = '0'

GZrLiq = '0'

GMoBCC = '-5800+2\*T'

GZrBCC = '-4250+2\*T'

GMoHCP = '-3800+2\*T'

GZrHCP = '-5280+2.9\*T'

L0Liq = '1512'

L0BCC = '6551'

L0HCP = '8981'

cLaves = [2, 1]

GMoLaves = GMoHCP

GZrLaves = GZrHCP

L0Laves = '-16488'

# PHASES

phases = []

phases.append(thermo.sol\_phase('liq', [GMoLiq, GZrLiq], [L0Liq]))

phases.append(thermo.sol\_phase(r'$BCC$', [GMoBCC, GZrBCC], [L0BCC]))

phases.append(thermo.sol\_phase(r'$HCP$', [GMoHCP, GZrHCP], [L0HCP]))

phases.append(thermo.stq\_phase(r'$Laves$', cLaves,

[GMoLaves, GZrLaves], [L0Laves]))

return phases