CME 570 HW4

Xiaoli Yan xyan11

November 14, 2019

1. **Determine the phase of the binary couple (e.g., both sides are bcc or fcc) and the composition difference between the couple. You should be able to determine the phase and composition range based on the phase diagram of the system. Plot the phase diagram and show that in your report. Also show the phase and composition range in the phase diagram.**

   I am using the Nickel-based demo databases, with Al-Cr as the binary alloy.

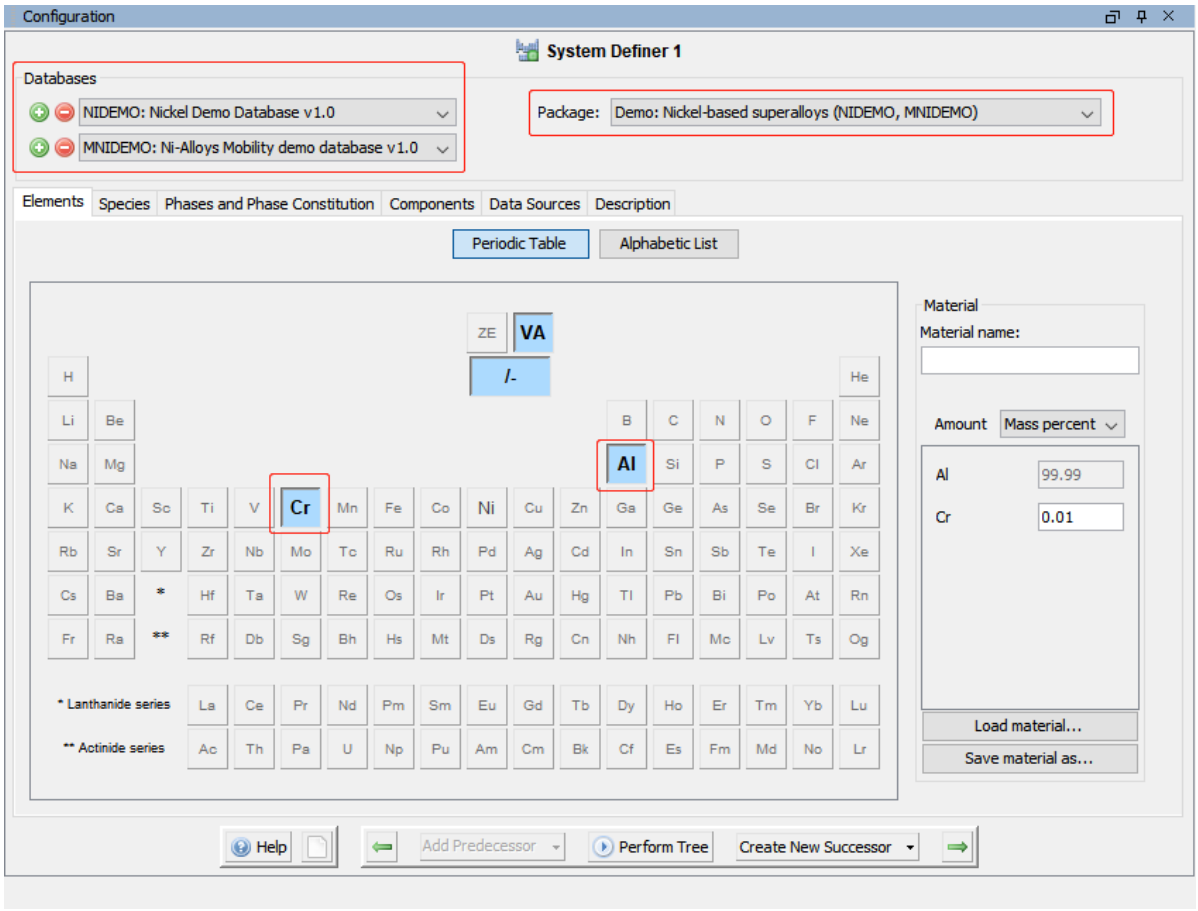

Figure 1: Thermodynamic and Mobility Data Sources

I only included the following phases: BCC_A2, FCC_A1, LIQUID, and GAS(not shown in phase diagram).

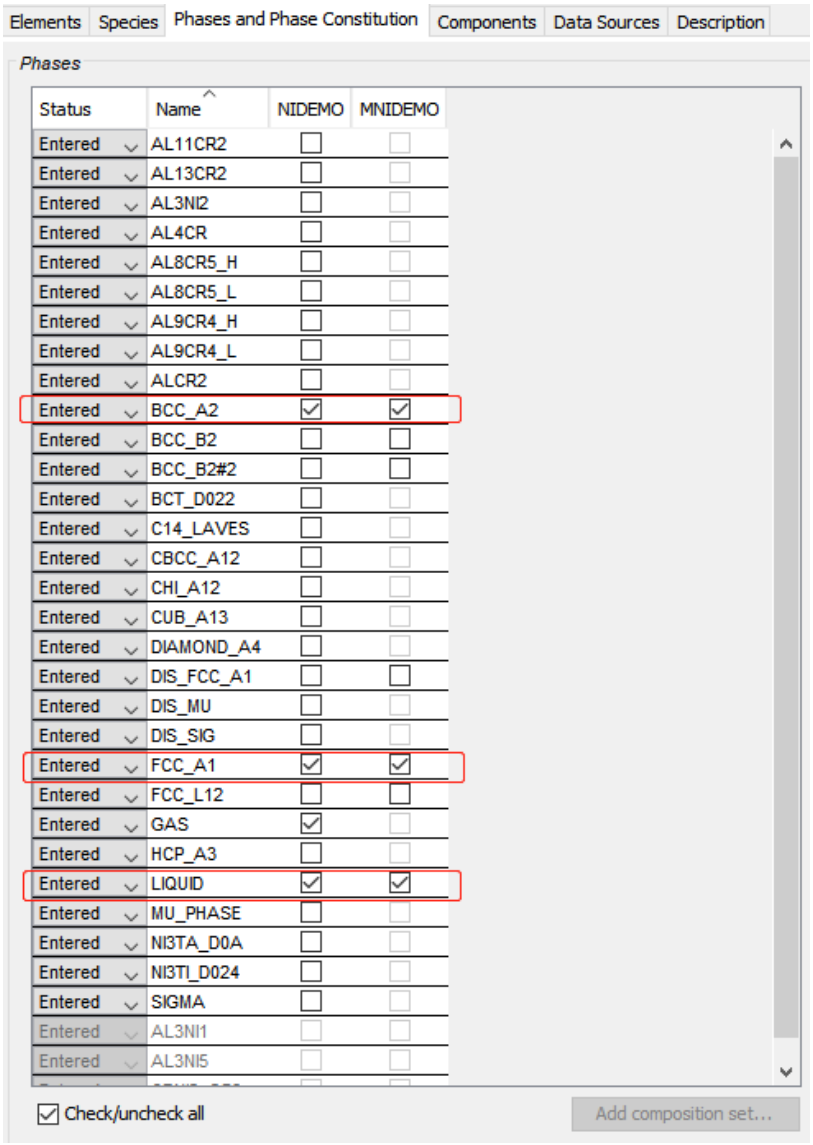

Figure 2: Included Phases in the Calculations

I ploted the phase diagram within $T \in [500\ K, 1250\ K]$ and $x_{Al} \in [0.00, 1.00]$.

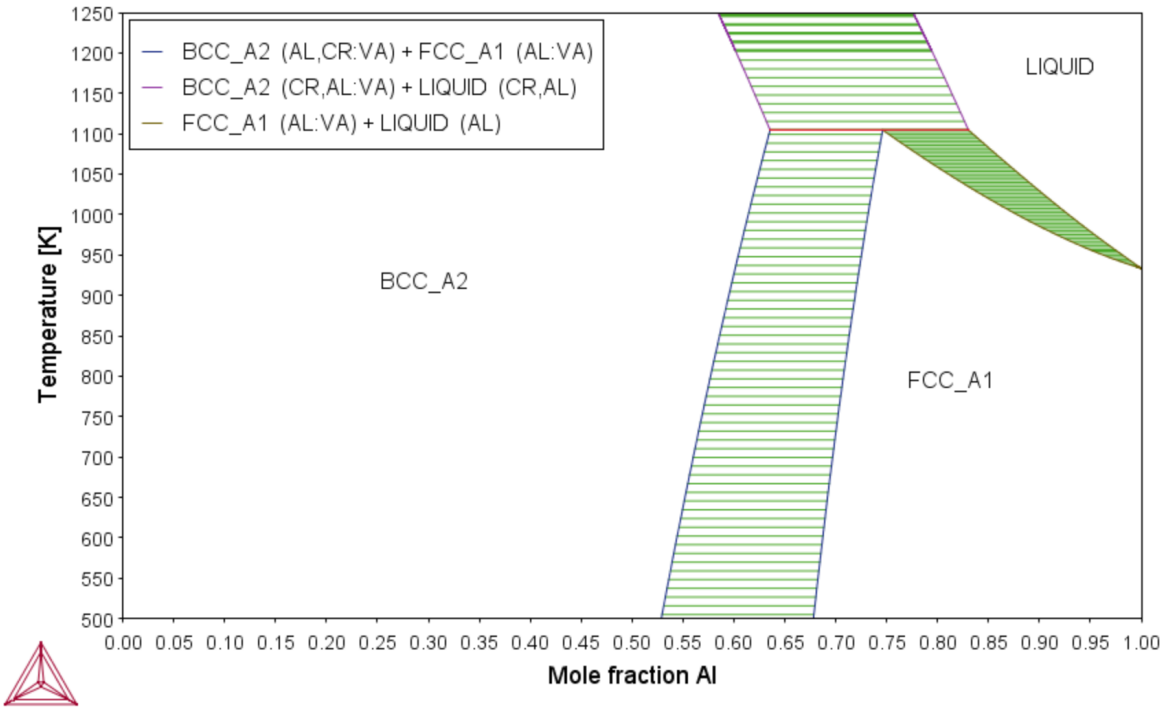

Figure 3: Phase Diagram of Al-Cr

Binary Couple at the following mole fraction compositions:
1. $x_{Al} = 0.8$, $x_{Cr} = 0.2$,
2. $x_{Al} = 1.0$, $x_{Cr} = 0.0$,
At $T = 900\ K$,
in the FCC_A1 region: (AL:VA=1:1).
Compositions of elements and vacuum are taken from the ThermoCalc "TCAL5" document[1].

2. **Assume that these two materials are joined together in a weldment and then kept at an elevated temperature for a period of time. Assume that the length of each side is 0.5 mm. Plot the following:**

   (a) **concentration profile**

   (b) **flux of each species**

   (c) **activity profile**

   (d) **chemical diffusion profile**

   You should be able to determine the simulation temperature based on the phase diagram.

   I configured the diffusion simulation as the following:

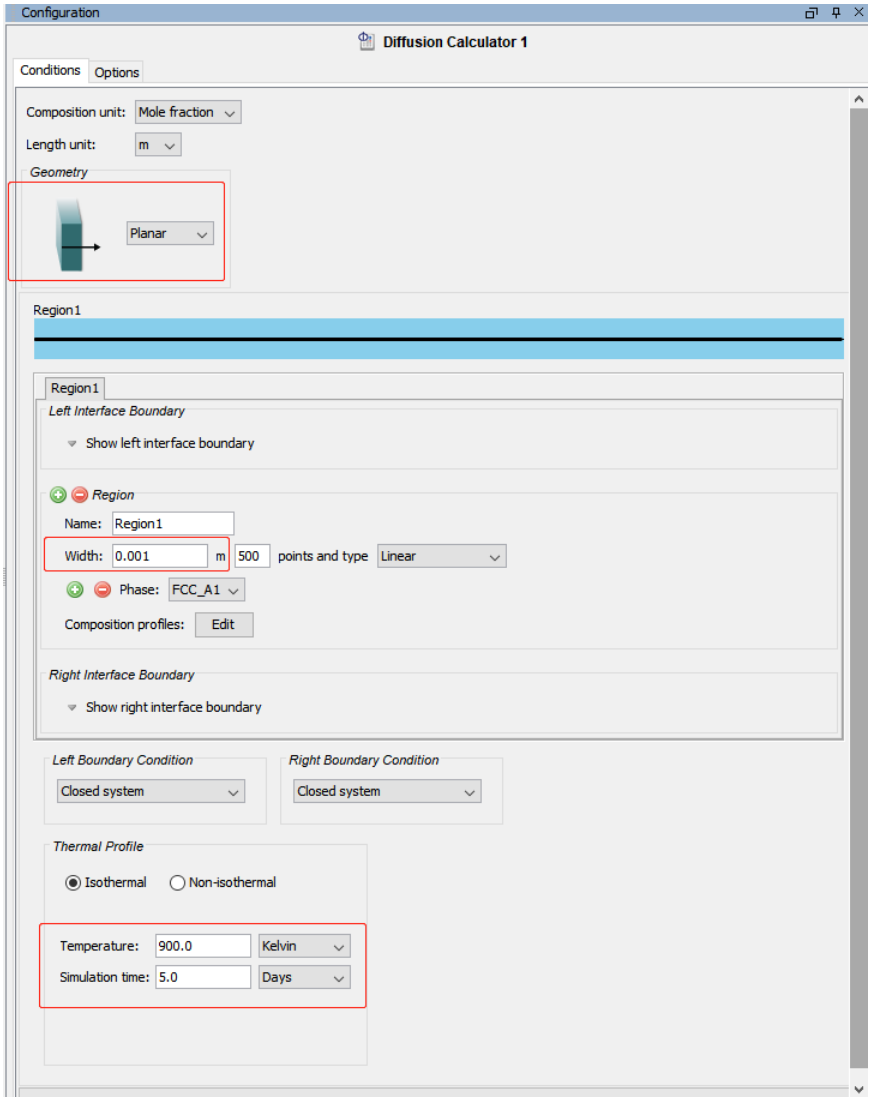

Figure 4: Simulation Configuration: FCC_A1, $T = 900$K

with the initial concentration profile as:

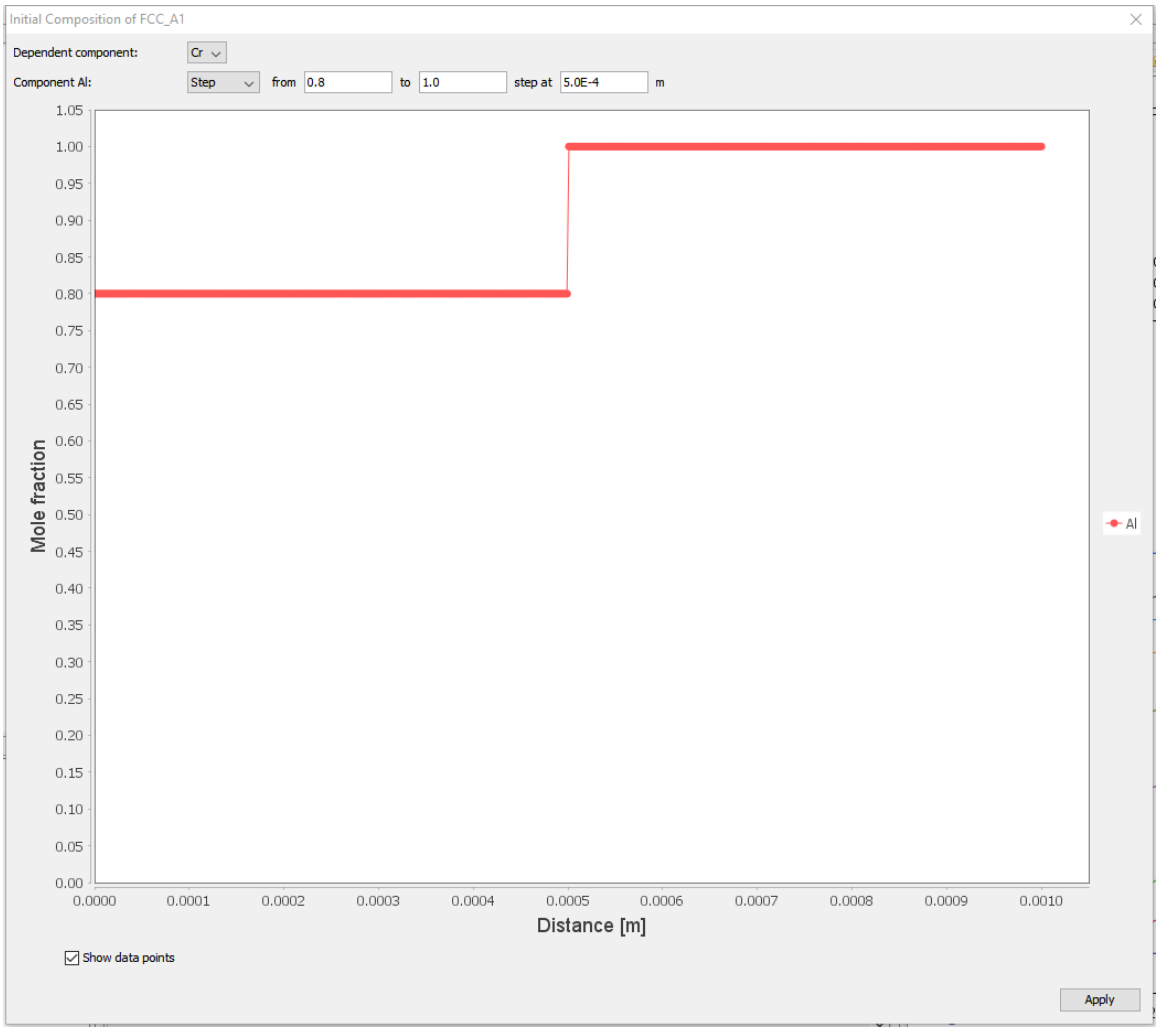

Figure 5: Initial Concentration Profile of Al

I set the solver to "homogenization" in the simulation options:

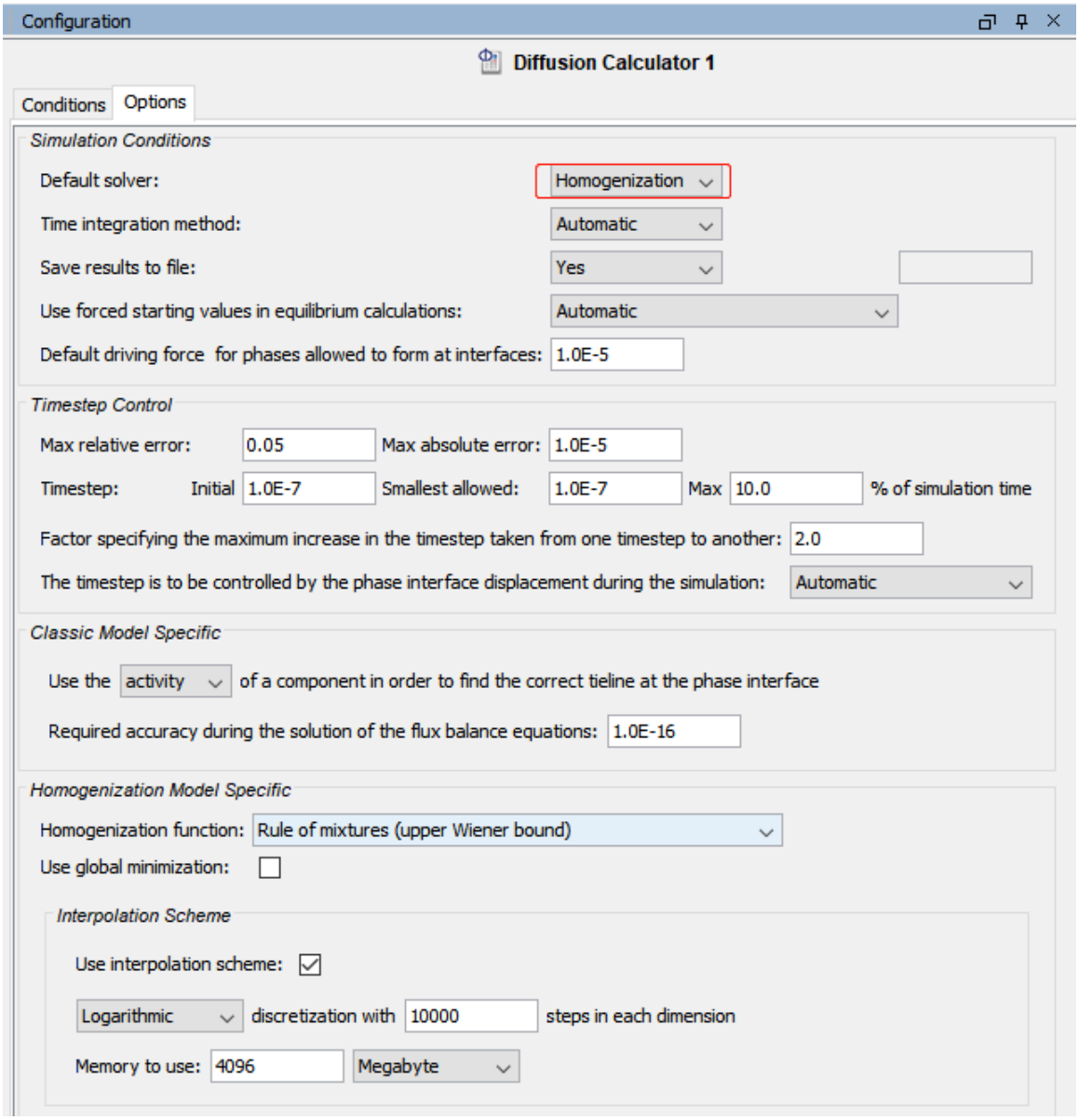

Figure 6: Simulation Options

(a) concentration profile

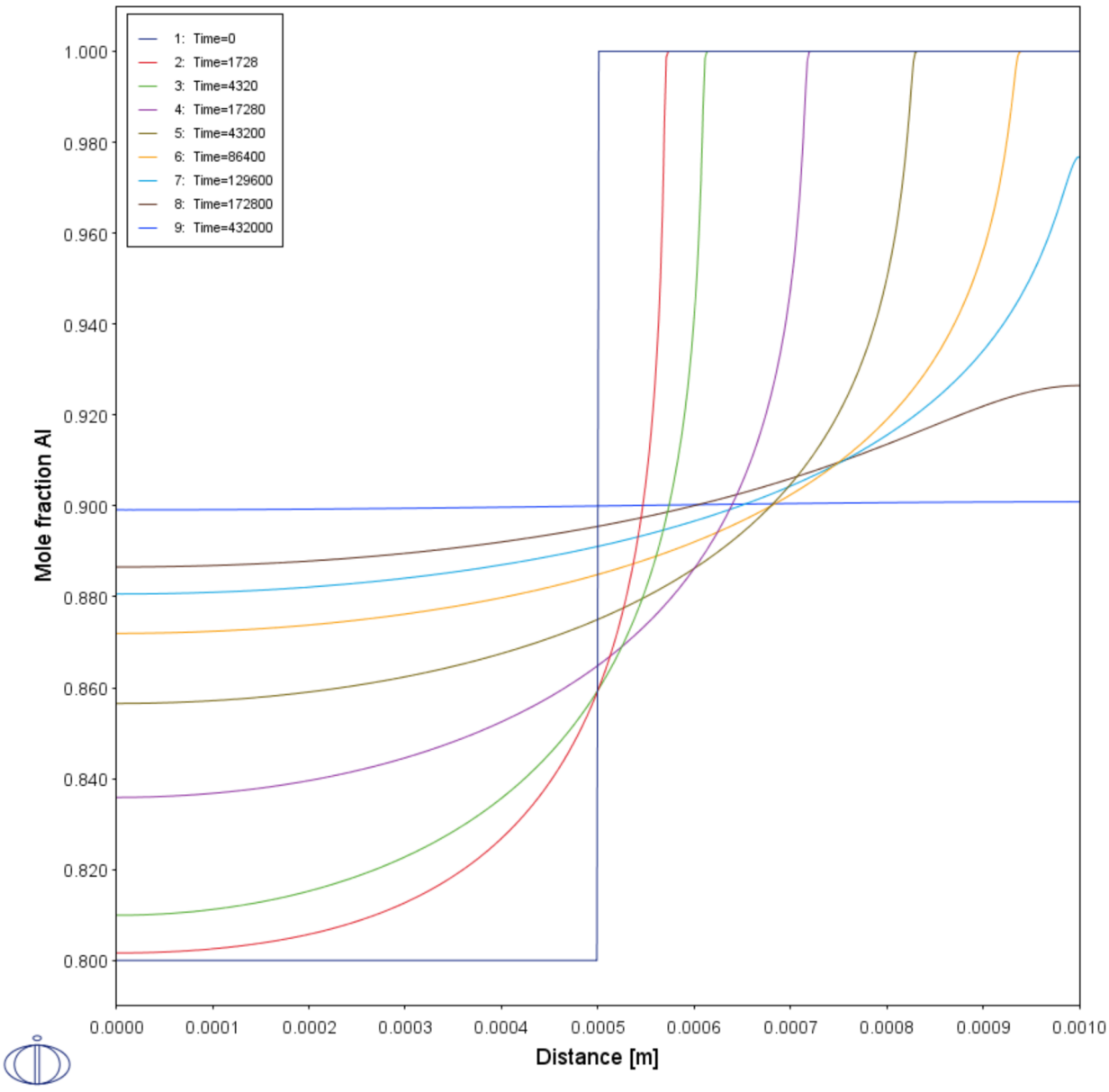

Figure 7: Composition Profile of the system at t=0, 0.02, 0.05, 0.2, 0.5, 1.0, 1.5, 2, 5 days

(b) flux of each species

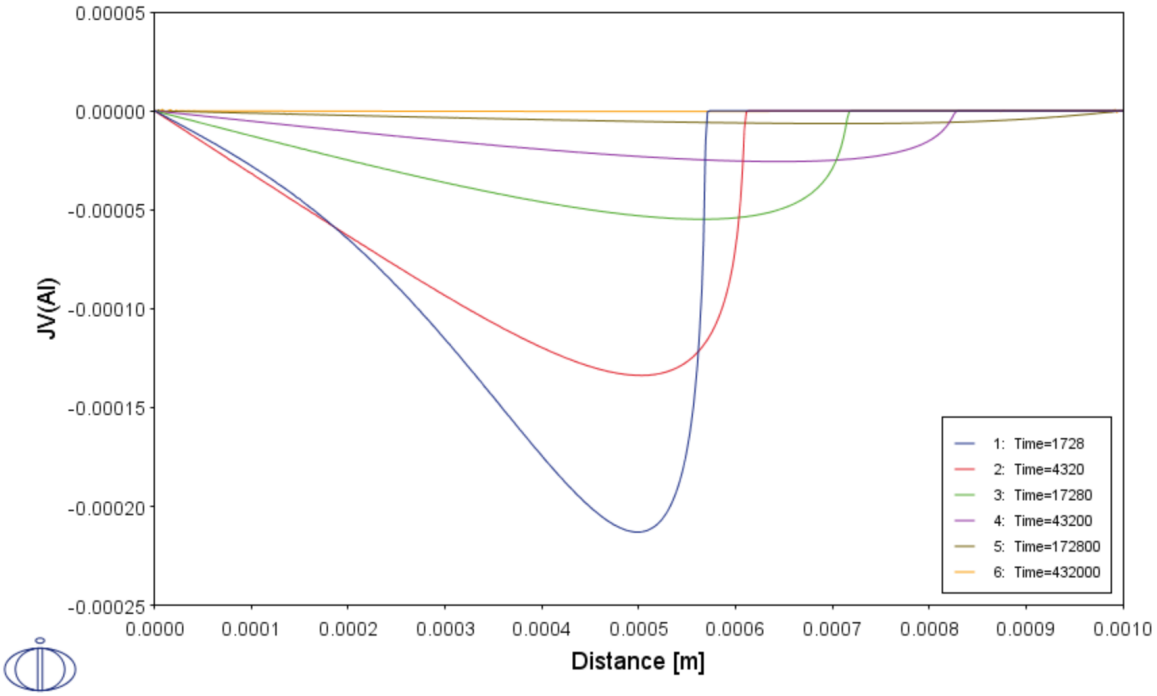

Figure 8: Al Flux Profile at t=0.02, 0.05, 0.2, 0.5, 2, 5 days

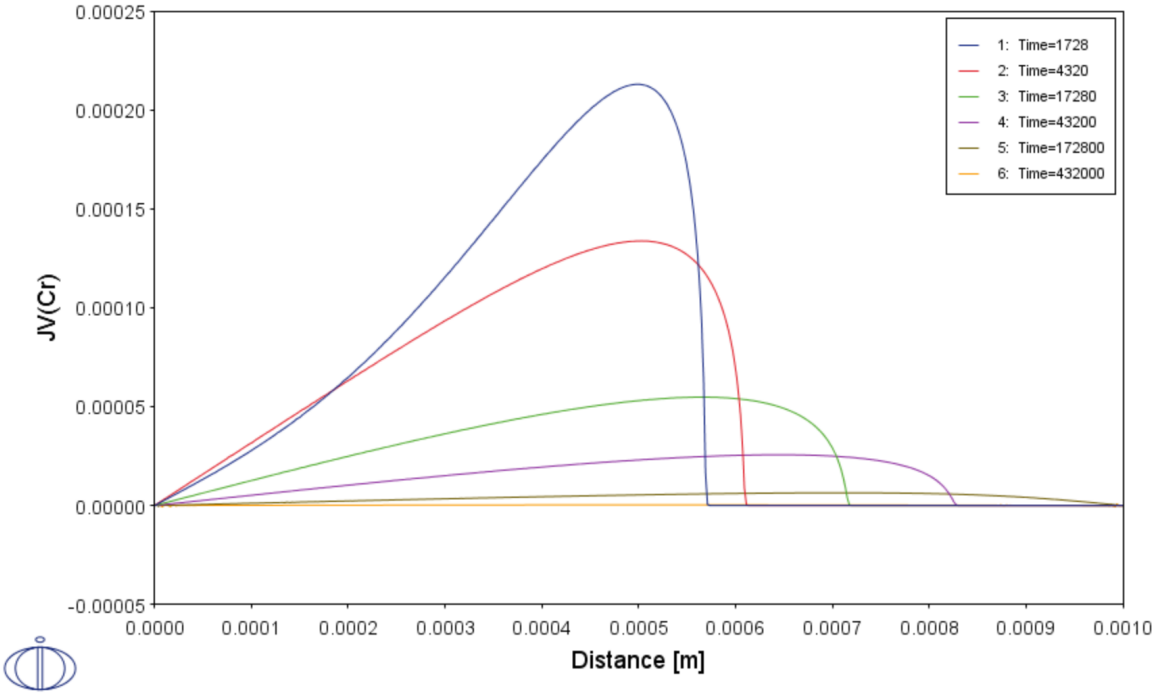

Figure 9: Cr Flux Profile at t=0.02, 0.05, 0.2, 0.5, 2, 5 days

(c) activity profile

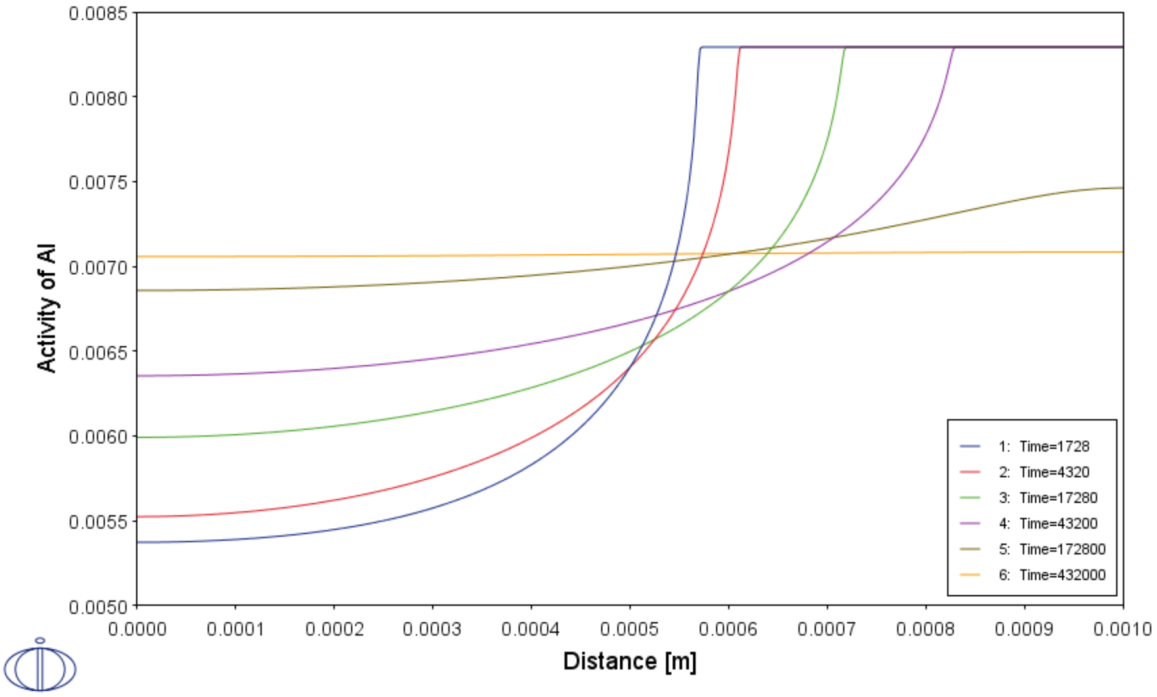

Figure 10: Al Activity Profile at t=0.02, 0.05, 0.2, 0.5, 2, 5 days

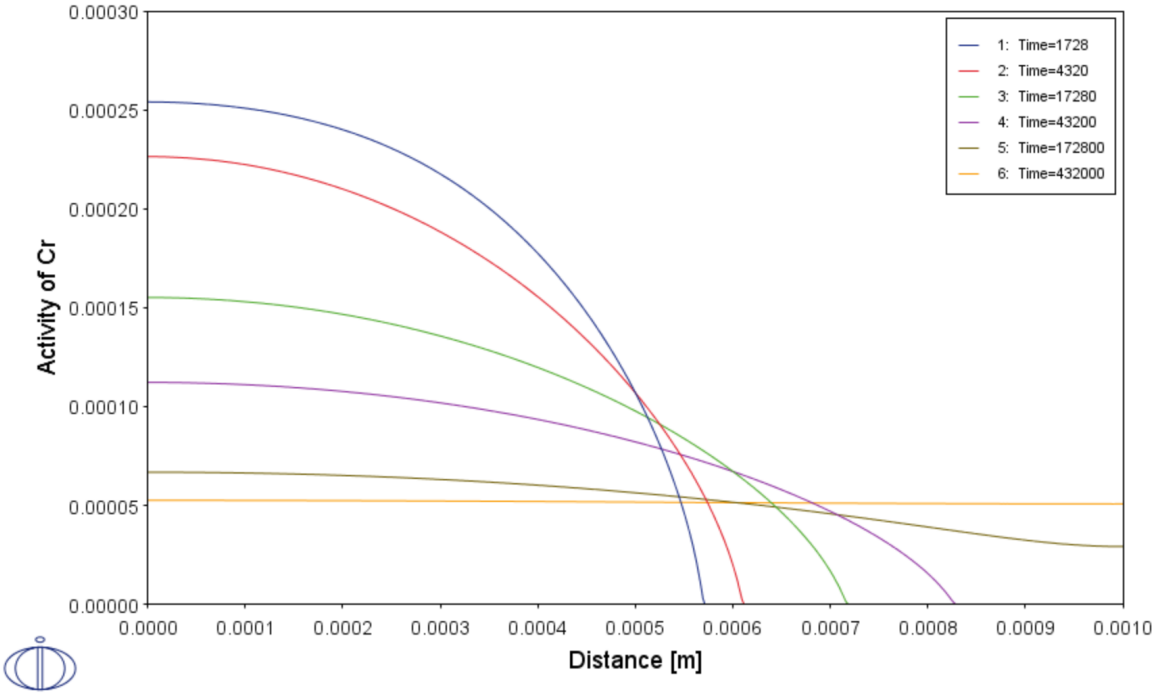

Figure 11: Cr Activity Profile at t=0.02, 0.05, 0.2, 0.5, 2, 5 days

(d) chemical diffusion profile

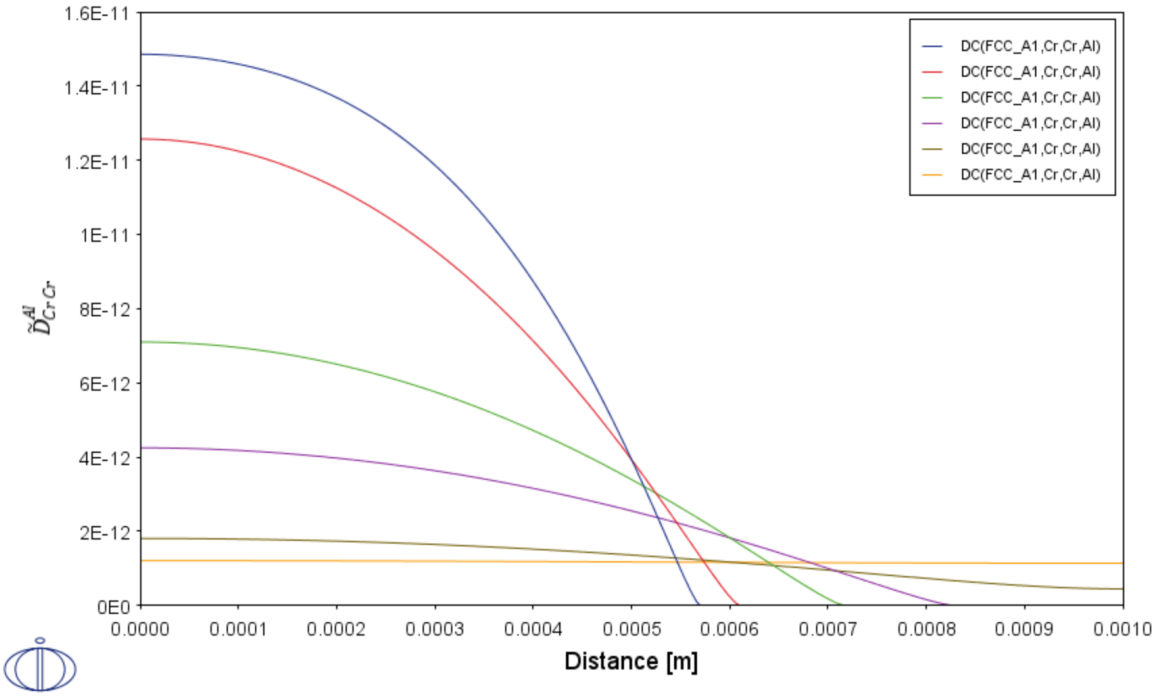

Figure 12: Al Chemical Diffusion Profile at t=0.02, 0.05, 0.2, 0.5, 2, 5 days

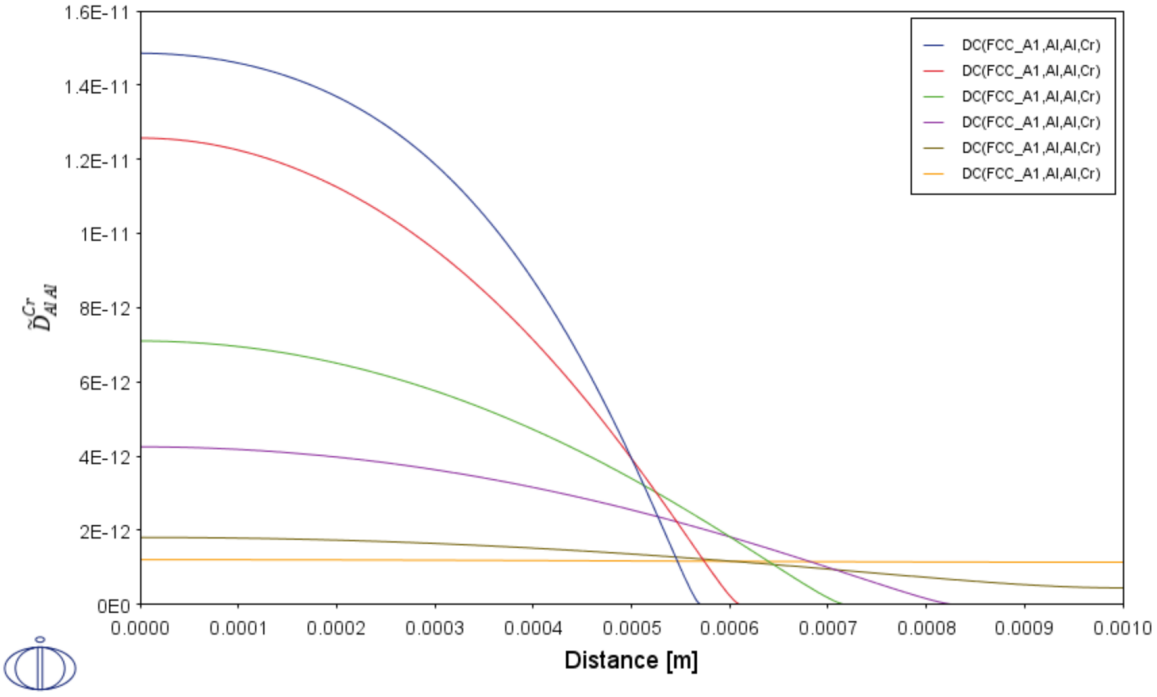

Figure 13: Cr Chemical Diffusion Profile at t=0.02, 0.05, 0.2, 0.5, 2, 5 days

3. **Having the concentration profile for a specific annealing time, use the Matano-Boltzmann relation to reproduce the chemical diffusivity for different concentration. Compare that with the results of your simulation.**

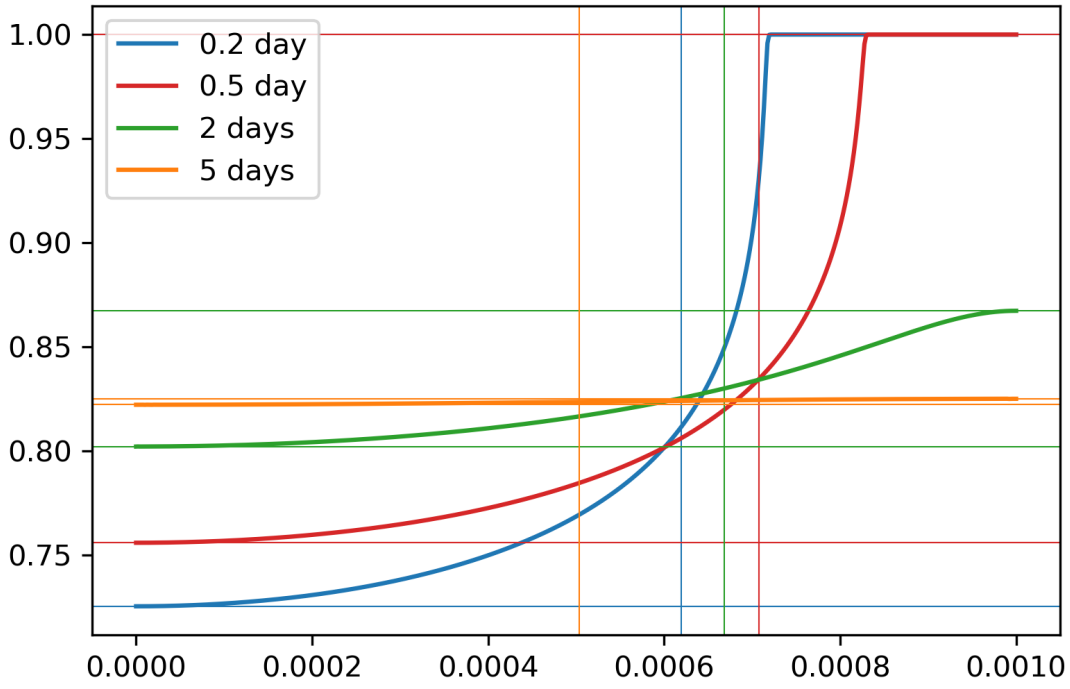

Figure 14: Al Concentration Profile and Matano Plane Positions at t=0.2, 0.5, 2, 5 days

At each annealing time: First I used the following formula to find the Matano plane position X_M:

$$X_M = \frac{1}{c(x=0.0) - c(x=0.001)} \int_{c(x=0.001)}^{c(x=0.0)} x dc \tag{1}$$

The code used to find Matano planes:

```
def matano(x, c):
    x = list(x)
    c = list(c)

    dc = [None] * (len(x)-1)

    sum_xdc = 0.0

    for i in range(0, len(dc)):
        dc[i] = c[i] - c[i-1]

        if dc[i] == 0:
            print(i)
            break
```

```
        sum_xdc = sum_xdc - (dc[i] * x[i+1])

    cL = c[0]
    cR = c[-1]

    x_M = sum_xdc / (cL - cR)

    return x_M

fig, ax = plt.subplots()

al = pd.read_csv("al_17280.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=17280'])
ax.plot(x, c, label="0.2 day", color='tab:blue')
x_M = matano(x, c)
ax.axvline(x=x_M, linewidth=0.5, color='tab:blue')
ax.axhline(y=c[0], linewidth=0.5, color='tab:blue')
ax.axhline(y=c[-1], linewidth=0.5, color='tab:blue')

al = pd.read_csv("al_43200.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=43200'])
ax.plot(x, c, label="0.5 day", color='tab:red')
x_M = matano(x, c)
ax.axvline(x=x_M, linewidth=0.5, color='tab:red')
ax.axhline(y=c[0], linewidth=0.5, color='tab:red')
ax.axhline(y=c[-1], linewidth=0.5, color='tab:red')

al = pd.read_csv("al_172800.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=172800'])
ax.plot(x, c, label="2 days", color='tab:green')
x_M = matano(x, c)
ax.axvline(x=x_M, linewidth=0.5, color='tab:green')
ax.axhline(y=c[0], linewidth=0.5, color='tab:green')
ax.axhline(y=c[-1], linewidth=0.5, color='tab:green')

al = pd.read_csv("al_432000.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=432000'])
ax.plot(x, c, label="5 days", color='tab:orange')
x_M = matano(x, c)
ax.axvline(x=x_M, linewidth=0.5, color='tab:orange')
ax.axhline(y=c[0], linewidth=0.5, color='tab:orange')
ax.axhline(y=c[-1], linewidth=0.5, color='tab:orange')
#ax.grid()
ax.legend()
fig.savefig("matano_planes.png", dpi=300)
plt.show()
```

Then I used the following formula to calculate the diffusivity:

$$D(c') = -\frac{1}{2t}\frac{\partial x}{\partial c}\bigg|_{c'} \int_{c_0}^{c'} x dc \tag{2}$$

where c_0 is the concentration at $x = 0.001$ m.

Python code used to plot and analyze:

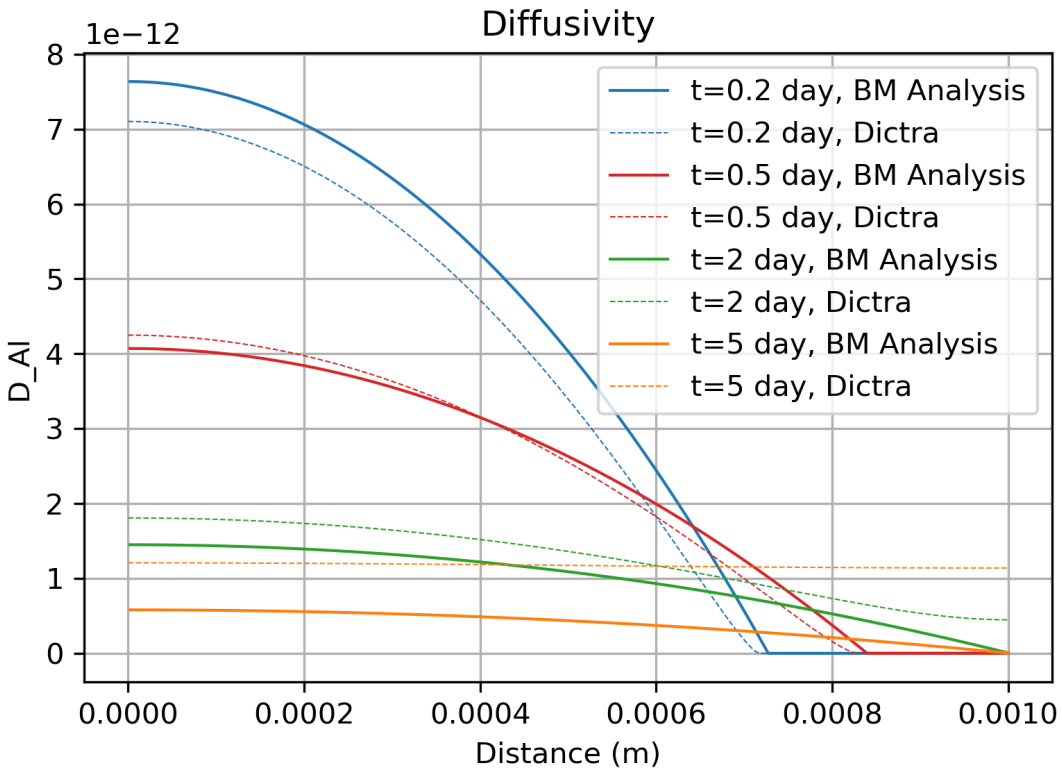

Figure 15: Al Chemical Diffusion Profile Comparison at t=0.2, 0.5, 2, 5 days

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

def dc_dx(x, c):
    x = list(x)
    c = list(c)
    dx = [None] * (len(x)-1)
    dc = [None] * (len(x)-1)
    cp = [None] * (len(x)-1)
    for i in range(0, len(x) - 1):
        dx[i] = x[i+1] - x[i]
        dc[i] = c[i+1] - c[i]
        cp[i] = dc[i] / dx[i]
    return x, c, cp

def int_x_dc(x, c, end_index):

    x = list(x)

    c = list(c)
```

```
    dc = [None] * (len(x)-1)
    int_x = [None] * (len(x))

    int_x[0] = 0.0
    max_index = end_index + 1
    for i in range(1, len(x)):
        int_x[i] = 0.0
        if i > end_index:
            int_x[i] = 0.0
        else:
            for j in range(i, end_index + 1):
                dc[i] = c[i] - c[i-1]
                int_x[i] = int_x[i] - (dc[i] * x[j])
    return int_x

fig, ax = plt.subplots()

t = 17280.0
al = pd.read_csv("al_17280.csv")

x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=17280'])
cd = [None] * (len(x) - 1)
end_index=len(c)-1
for j in range(0, len(c)):
    if c[j] == 1.0:
        end_index = j
        #print(j)
        break
int_x = int_x_dc(al['Distance [m]'], al['1: Time=17280'], end_index)
for i in range(0, (len(x) - 1)):
    if cp[i] == 0.0:
        cd[i] = 0.0
    else:
        cd[i] = -int_x[i+1] / cp[i] / (2.0 * t)

ax.plot(x[1:], cd, label="t=0.2 day, BM Analysis", linewidth=1.0, color='tab:blue')
diff = pd.read_csv("diff17280.csv")
ax.plot(x, list(diff['1: Time=17280']), '--', label="t=0.2 day, Dictra", linewidth=0.5,
    color='tab:blue')


t = 43200.0
al = pd.read_csv("al_43200.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=43200'])
end_index=len(c)-1
for j in range(0, len(c)):
    if c[j] == 1.0:
        end_index = j
        #print(j)
        break
int_x = int_x_dc(al['Distance [m]'], al['1: Time=43200'], end_index)
for i in range(0, (len(x) - 1)):
    if cp[i] == 0.0:
        cd[i] = 0.0
```

```
    else:
        cd[i] = -int_x[i+1] / cp[i] / (2.0 * t)
ax.plot(x[1:], cd, label="t=0.5 day, BM Analysis", linewidth=1.0, color='tab:red')
diff = pd.read_csv("diff43200.csv")
ax.plot(x, list(diff['1: Time=43200']), '--', label="t=0.5 day, Dictra", linewidth=0.5,
    color='tab:red')

t = 172800.0
al= pd.read_csv("al_172800.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=172800'])
end_index=len(c)-1
for j in range(0, len(c)):
    if c[j] == 1.0:
        end_index = j
        #print(j)
        break
int_x = int_x_dc(al['Distance [m]'], al['1: Time=172800'], end_index)
for i in range(0, (len(x) - 1)):
    if cp[i] == 0.0:
        cd[i] = 0.0
    else:
        cd[i] = -int_x[i+1] / cp[i] / (2.0 * t)
ax.plot(x[1:], cd, label="t=2 day, BM Analysis", linewidth=1.0, color='tab:green')
diff = pd.read_csv("diff172800.csv")
ax.plot(x, list(diff['1: Time=172800']), '--', label="t=2 day, Dictra", linewidth=0.5,
    color='tab:green')


t = 432000.0
al = pd.read_csv("al_432000.csv")
x, c, cp = dc_dx(al['Distance [m]'], al['1: Time=432000'])
end_index=len(c)-1
for j in range(0, len(c)):
    if c[j] == 1.0:
        end_index = j
        #print(j)
        break
int_x = int_x_dc(al['Distance [m]'], al['1: Time=432000'], end_index)
for i in range(0, (len(x) - 1)):
    if cp[i] == 0.0:
        cd[i] = 0.0
    else:
        cd[i] = -int_x[i+1] / cp[i] / (2.0 * t)
ax.plot(x[1:], cd, label="t=5 day, BM Analysis", linewidth=1.0, color='tab:orange')
diff = pd.read_csv("diff432000.csv")
ax.plot(x, list(diff['1: Time=432000']), '--', label="t=5 day, Dictra", linewidth=0.5,
    color='tab:orange')

ax.set(xlabel='Distance (m)', ylabel='D_Al', title='Diffusivity')
ax.grid()
ax.legend()
fig.savefig("D_Al.png", dpi=300)
plt.show()
```

4. **Examine Darken's equation based in the calculated tracer diffusivity, thermodynamic factor, and chemical diffusivity.**

$$\tilde{D} = (x_B D_A^{Tr} + x_A D_B^{Tr})\varphi \tag{3}$$

x_A, x_B are the mole fractions of atom types A and B; D_A^{Tr}, D_B^{Tr} are the tracer diffusivities of atom types A and B; φ is the thermodynamic factor.

During the calculation, I am using the plots at same simulation condition and annealing time at 2 days.

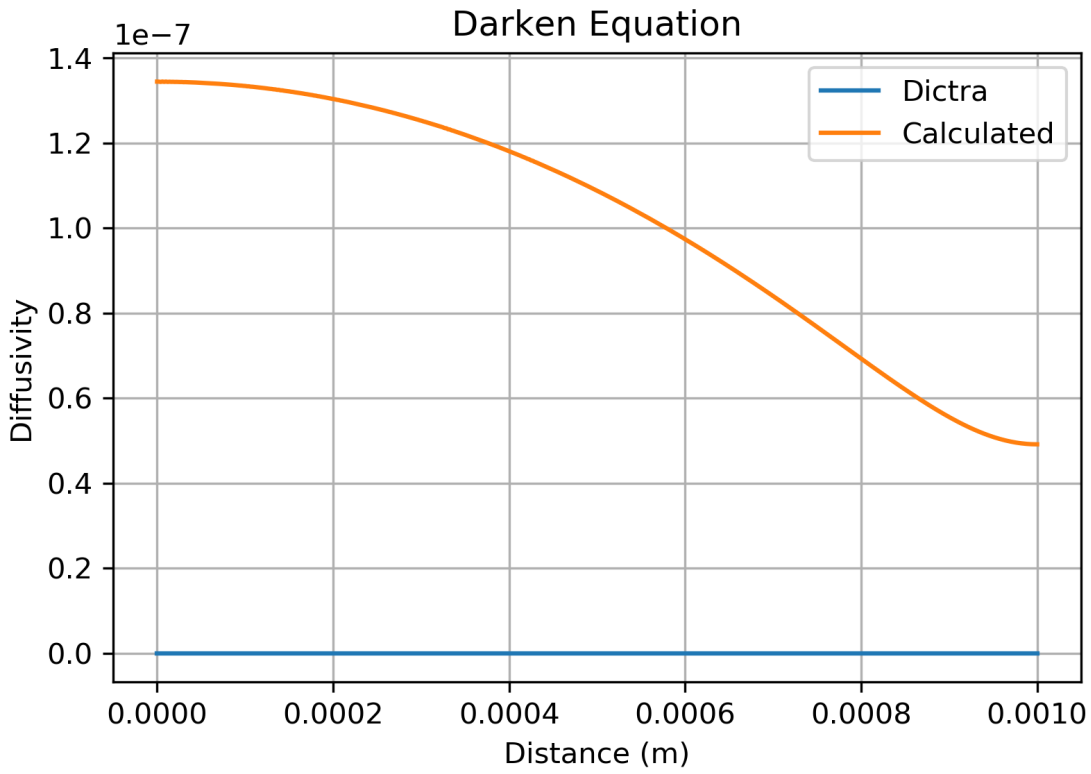

Figure 16: Dictra Calculated Diffusivity and Darken Calculated Difusivity

The plots are not in the same order of magnitude, but after I removed the thermodynamic factor,

$$\tilde{D} = x_B D_A^{Tr} + x_A D_B^{Tr} \tag{4}$$

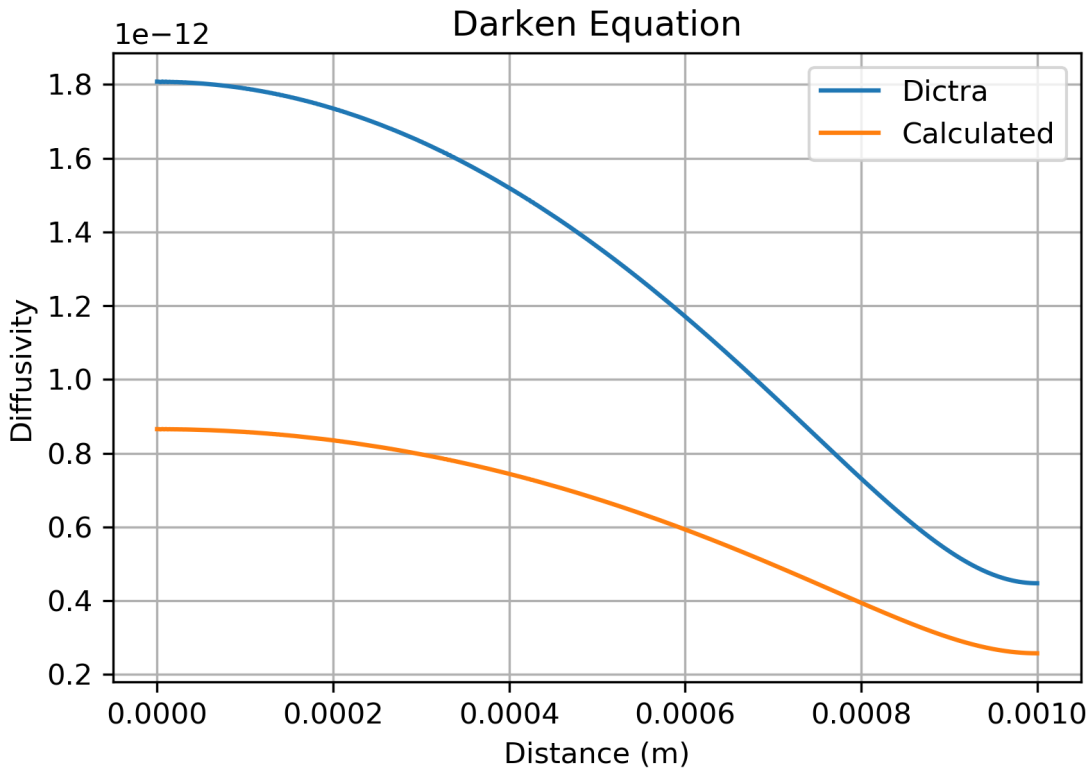

Figure 17: Dictra Calculated Diffusivity and Darken Calculated Difusivity

References

[1] Thermocalc.com. [online] Available at: `https://www.thermocalc.com/media/19849/tcal5_extended_info.pdf`[Accessed 9 Nov. 2019].