Materials Science and Engineering Problems

MSE Faculty

September 14, 2016

This document includes the computational exercises that are assigned throughout the MSE core curriculum. In general, these require the use of MATLAB.

Contents

314 Computational Exercises

1. (2015) For this problem, you will be using MATLAB's symbolic solver (fzero) and function handles to find the zero of an equation. On last week's quiz we found that 89.1 grams of ice were necessary to cool a 1 kg block of Pb down to 300 K from 600 K. We will be plotting the change in temperature for both the Pb and ice. You will need the following parameters:

$$C_p^{Pb} = 0.1169 + 4.2x10^{-5}T \, \frac{\text{kg}}{\text{kg}\cdot\text{K}}$$
$$C_p^{H_2O} = 4.2 \, \frac{\text{kJ}}{\text{kg}\cdot\text{K}}$$
$$\Delta H_{fus}^{H_2O} = 344 \, \frac{\text{kJ}}{\text{kg}}$$
$$\frac{dH}{dt} = a\Delta T$$

Here a is a coefficient that controls the heat transfer in conduction, which we will assume to be $3.33x10^{-3}$ kJ/K · s, andΔT is the temperature difference between the two materials.

(a) Use the equations derived in class to plot the change in temperature for H2O and Pb. Assume dt=1 s and calculate the first 200 time steps. How do you know when the system is at equilibrium?

(b) Create a plot that shows the amount of water in the system as a function of time. At what time is all the ice gone?

(c) Create a plot that shows the total heat transfer occurring between the Pb and H2O. How can you tell when equilibrium is reached from this plot?

2. (2015) We will be putting together a program to help calculate phase diagrams of all sorts piece by piece. The first step is to create a MATLAB script that solves a system of two equations. The system is below:

$$G_s - X_s\frac{dG_s}{dX_s} = G_\ell - X_\ell\frac{dG_\ell}{dX_\ell} \tag{0.1}$$

$$\frac{dG_s}{dX_s} = \frac{dG_\ell}{dX_\ell} \tag{0.2}$$

where G_s and G_ℓ are given by the following expressions:

$$G_s\left(X_s\right) = \Omega_s X_s\left(1-X_s\right) + RT\left[X_s \ln X_s + \left(1-X_s\right)\ln\left(1-X_s\right) + 200X_s - 400\left(1-X_s\right)\right].$$

$$G_\ell\left(X_\ell\right) = \Omega_\ell X_\ell\left(1-X_\ell\right) + RT\left[X_\ell \ln X_\ell + \left(1-X_\ell\right)\ln\left(1-X_\ell\right)\right].$$

Here G_s is the Gibbs energy of the solid phase, G_ℓ the Gibbs free energy of the liquid phase, R the gas constant (8.314 J/K), T the absolute temperature, X_s and X_ℓ are the compositions of the solid and liquid phase respectively, and Ω_s, Ω_ℓ are parameters to be defined later. We can rewrite Eqs. 0.1 and 0.2 as follows:

$$G_s - X_s\frac{dG_s}{dX_s} - G_\ell + X_\ell\frac{dG_\ell}{dX_\ell} = 0 \tag{0.3}$$

$$\frac{dG_s}{dX_s} - \frac{dG_\ell}{dX_\ell} = 0 \tag{0.4}$$

Create a MATLAB function that takes X_s, X_ℓ, T, Ω_s and Ω_ℓ as inputs and then create a script that uses the MATLAB command fsolve to calculate X_s and X_ℓ for T=700K, $\Omega_\ell = 1500\,\text{cal/mol}$ and $\Omega_s = 3000\,\text{cal/mol}$.

3. (2015) Now that we are able to solve for the composition of the solid and liquid at one point, we will improve our script to calculate it over a range of temperatures. Start with an initial guess for both the solid and liquid near zero and a temperature of 900 K. Determine the composition of the liquid and solid down to 1 K for each temperature using a for loop. Make sure to update your guess with the correct answer for the previous temperature to help your program converge. Repeat again starting from 600 down to 1 K, this time starting with an initial guess near 1, and plot your results. Use the following parameters to make the Gibbs energy more physical:

$$T_m^\alpha = 900\,\mathrm{K}$$

$$T_m^\beta = 600\,\mathrm{K}$$

$$\Delta H_f^\alpha = 2000\,\frac{\mathrm{cal}}{\mathrm{mol}}$$

$$\Delta H_f^\beta = 1300\,\frac{\mathrm{cal}}{\mathrm{mol}}$$

The liquid and solid free energies are given by the following expressions. (Note that the Gibbs energy for the solid phase has changed slightly and should be adjusted in your code. In these units $R = 1.987\,\mathrm{cal/mol \cdot K}$

$$G_s\left(X_s\right) = \Omega_s X_s \left(1 - X_s\right) + RT\left[X_s \ln X_s + \left(1 - X_s\right)\ln\left(1 - X_s\right) + X_s \Delta G_\beta^{\ell \to s} + \left(1 - X_s\right)\Delta G_\alpha^{\ell \to s}\right].$$

$$G_\ell\left(X_\ell\right) = \Omega_\ell X_\ell \left(1 - X_\ell\right) + RT\left[X_\ell \ln X_\ell + \left(1 - X_\ell\right)\ln\left(1 - X_\ell\right)\right].$$

As a reminder, the free energy change for the melting transition can be written in terms of the enthalpic and entropic contributions to the free energy:

$$\Delta G^{s \to \ell} = \Delta H^{s \to \ell} - T\Delta S^{s \to \ell}$$

4. (2015) Our phase diagram calculation is almost complete! We only have to find the equilibrium between the two solid phases left. To do that, we simply take the derivative of the Gibbs free energy of the solid phase and set it equal to zero. The equation becomes:

$$RT\ln\left(\frac{X_s}{1 - X_s}\right) - \Omega_s\left(2X_s - 1\right)$$

Again, assume that Ω_s= 3000 cal/mol. This is easily done by creating a for loop that solves for the temperature at each composition between .01 and .99. Plot your results on the same figure from the previous homework and voila, your first phase diagram!

Now with your working code, replot the diagrams for the following interaction coefficients. You will have to change the range of temperatures for one of the sets below. You can figure out which one it is, if you think of the physical significance of the parameters.

(a) $\Omega_s = 3000\,\mathrm{cal/mol} : \Omega_\ell = 0$

(b) $\Omega_s = 0 :\; \Omega_\ell = 3000\,\mathrm{cal/mol}$

(c) $\Omega_s = 0;\, \Omega_\ell = 0$

(d) $\Omega_s = 3000\,\mathrm{cal/mol} : \Omega_\ell = 3000\,\mathrm{cal/mol}$

For each phase diagram, plot your results and describe how the changing interaction parameters changed the shape of the plot.

316-1 Simulation Exercise: Monte Carlo Simulation of Decomposition in a Binary Alloy

Background

Scientific problem

We want to analyze the thermodynamic evolution of a A-B alloy by simulation. We assume that this system has the phase diagram presented in Figure 0.1. In this figure we see that for temperatures lower than T_C, the A-B alloy decomposes in two phases α and β with equilibrium concentrations X_B^{α} and X_B^{β}. The experiment that we want to model involves the following steps:

1. We mix together the same number of moles of elements A and B to obtain a homogeneous alloy at some temperature above T_c.
2. The temperature is reduced to T_0.
3. The temperature is held fixed at T_0, and the system evolves to form two different phases, with compositions X_B^{α} and X_B^{β}.

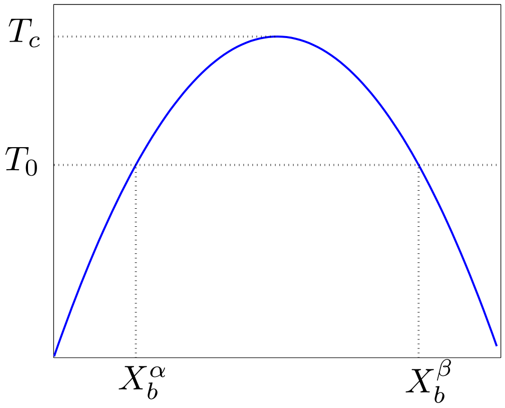

Figure 0.1: A-B alloy phase diagram

Atomistic Monte Carlo Model

In this section, we introduce the Atomistic Monte Carlo model that we will use to model the decomposition of the A-B alloy.

Atomistic model

In this model, we suppose that the two elements (A and B) have the same lattice structure. This lattice is represented by a matrix with periodic boundary conditions on its edges (see Figure 0.2). In 2 dimensions the left edge is connected to the right edge and the upper edge is connected to the lower edge. We reproduce the system evolution at the atomistic level: vacancies present in the lattice migrate from site to site by exchanging their position with their first nearest neighbors. The successive displacements of vacancies make the system evolve toward its equilibrium state.

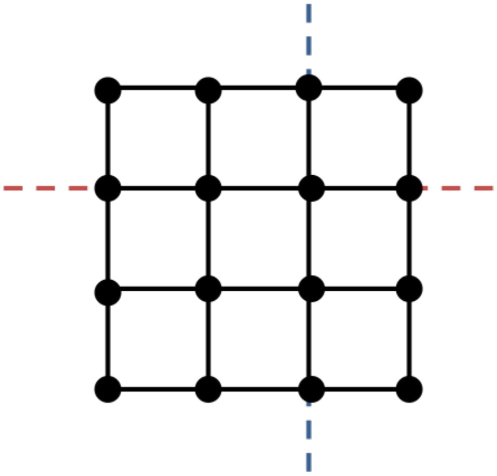

Figure 0.2: Lattice with periodic boundary conditions. The blue and red dashed lines represent bonds between sites induced by periodic boundary conditions.

Monte Carlo model

The thermodynamic evolution of the alloy is modeled with a Monte Carlo process. The principle of Monte Carlo simulations is to model the A-B alloy evolution in a statistic way. To understand this model we can consider individual jumps of a vacancy into one of the z nearest neighbor positions. Within a certain specified time step, Δt, these different possible jumps occur with a probability Γ_μ where μ is an index that indicates which direction the vacancy will move. In a simple cubic lattice, for example, $z = 6$, and the 6 values of μ correspond to jumps in the positive and negative x, y and z directions. The sum over all possible jump probabilities in the statistical time must sum to 1:

$$\sum_{\mu=1}^{z} \Gamma_\mu = 1 \tag{0.5}$$

To figure out which direction the vacancy moves, we draw a random number r_n between 0 and 1. The jump performed by the system during the time Δt is the k^{th} one such that the following condition holds:

$$\sum_{\mu=1}^{k-1} \Gamma_\mu < r_n \leq \sum_{\mu=1}^{k} \Gamma_\mu \tag{0.6}$$

Probabilities of transitions Γ_μ are related to the energetic barrier associated with vacancy motion, which we refer to as ΔE_μ. Because vacancy hopping is a thermally activated process, we can use an Arrhenius rate expression:

$$\Gamma_\mu = \Gamma_0 \exp\left(-\frac{\Delta E_\mu}{k_B T_0}\right) \tag{0.7}$$

where Γ_0 is a constant, k_B is Boltzmann's constant and T_0 is the temperature of the system.

The energy barrier is the difference between the maximum energy of the system during the jump (the position of the migrating atom at this maximum energy is called the saddle point) and the energy of the system before the jump.

$$\Delta E_\mu = E^{SP} - E^{ini} \tag{0.8}$$

Here the superscript *SP* refers to 'Saddle Point' and *ini* means 'initial', as shown in Figure 0.3.

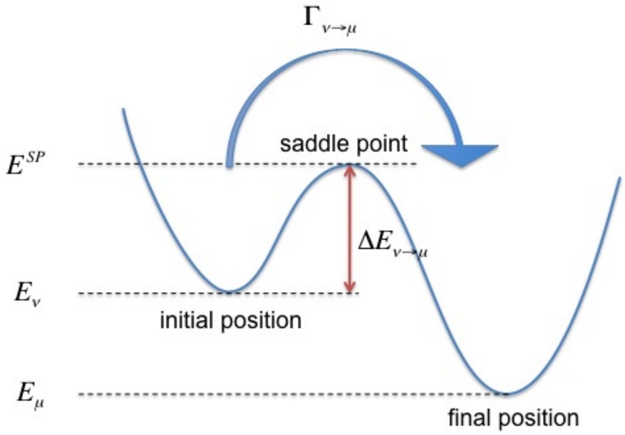

Figure 0.3: Schematic representation of the evolution of the system energy during a single atomic jump.

Energetic model

To compute the energy barriers of the different possible jumps ΔE_μ, we have to use an energetic model. In Monte Carlo simulations, we usually use an Ising model or Broken bond model. In this energetic model, we assume that the total energy of the system is equal to the sum of interaction energies ε_{ij} between the different elements (atoms of type A and B and vacancies V) placed on the lattice sites.

$$E_\nu = \Sigma_{ij}\varepsilon_{ij} \tag{0.9}$$

With this energetic model, the migration barrier of an exchange between an element X and the vacancy V becomes:

$$\Delta E_\nu = \sum_k \varepsilon^{SP}_{Xk} - \sum_i \varepsilon_{Xi} - \sum_j \varepsilon_{Vj} \tag{0.10}$$

where ε^{SP}_{Ak} are interaction energies between the atom migrating and its neighbors at the saddle point, ε_{Ai} are interaction energies between the atom migrating and its neighbors before the jump and ε_{Vj} are interaction energies between the vacancy and its neighbors before the jump. The indices i, j and k indicate the following neighbors:

Index	Meaning
i	nearest neighbors of the migrating atom before the jump
j	nearest neighbors of the vacancy before the jump
k	nearest neighbors of migrating atom at the saddle point

In theory, the range of interaction distances between elements are unlimited. In practice, we usually restrict these interactions to first and sometimes second nearest neighbors.

For example, the system presented in figure 0.4 has:

- 3 A-V interactions
- 1 B-V interaction
- 3 A-B interactions
- 9 A-A interactions

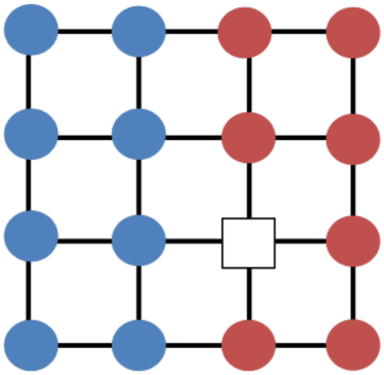

Figure 0.4: Example of configuration of atoms in the lattice. The red circles are A atoms and the blue circles are B atoms. The square is the vacancy

- 12 B-B interactions

Therefore, $E_v = 3\varepsilon_{AV} + 1\varepsilon_{BV} + 3\varepsilon_{AB} + 9\varepsilon_{AA} + 12\varepsilon_{BB}$. If we suppose that the vacancy exchange its position with the B atom on its left side, the configuration of the system at the saddle point is the one presented in figure 0.5.

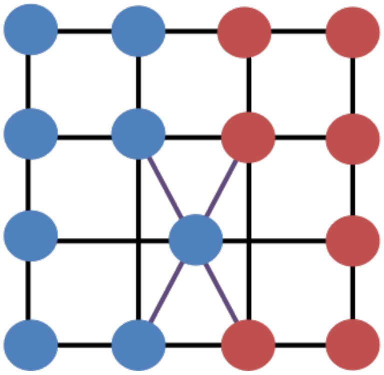

Figure 0.5: Configuration of the system at the saddle point if the vacancy echange its position with the B atom on its left

In this configuration, the system has:

- 2 B-B interactions at the saddle point
- 2 A-B ineractions at the saddle point
- 3 A-B interactions
- 9 A-A interactions
- 9 B-B interactions

so $E^{SP} = 2\varepsilon^{SP}_{BB} + 2\varepsilon^{SP}_{BA} + 3\varepsilon_{AB} + 9\varepsilon_{AA} + 9\varepsilon_{BB}$. The migration barrier of this jump is therefore:

$$\Delta E_v = 2\varepsilon^{SP}_{BB} + 2\varepsilon^{SP}_{BA} - 3\varepsilon_{AV} - 1\varepsilon_{BV} - 3\varepsilon_{BB}$$

Modeling of scientific problem

Here we assume that the two elements A and B have the same simple cubic lattice. We model the A-B alloy as a matrix in 2D with nx rows and ny columns and with periodic boundary conditions on its edges. To simplify the problem, we introduce only one vacancy in the lattice (so 1 vacancy for $nx \times ny$ sites), initially located in the middle of the matrix. As we only interest ourselves to the thermodynamic evolution of the system (and not to its kinetic evolution), we assume that the alloy evolves with normalized time steps of 1 until a maximum time t_{max}. At each time step, the vacancy exchanges its position with one of its neighbors.

To simplify the energetic model we suppose that the sum of interaction energies between the atom migrating and its neighbors at the saddle point $\sum_k \varepsilon_{Xk}^{SP}$ is a constant equal to $3\,eV$. In addition, we suppose that $\varepsilon_{AA} = \varepsilon_{BB} = \varepsilon_{AV} = \varepsilon_{BV} = 0\,eV$. The only interaction which can be different from zero is thus ε_{AB}.

The free enthalpy of the alloy is expressed by

$$\Delta G_{mix} = \Omega X_A X_B - T\Delta S_{mix} \tag{0.11}$$

with Ω the ordering energy of the alloy and ΔS_{mix} the configurational entropy of mixing of the alloy given by :

$$\Delta S_{mix} = -k_B \left[X_A \ln X_A + X_B \ln X_B\right] \tag{0.12}$$

For a symmetrical miscibility gap, the ordering energy is

$$\Omega = 2k_B T_C \tag{0.13}$$

where T_C is the critical temperature of the miscibility gap ($T_C = 1000\,K$ in this study). In broken bond models with only first nearest neighbors interactions we have:

$$\Omega = z\left(\epsilon_{AB} - \frac{1}{2}\left(\varepsilon_{AA} + \varepsilon_{BB}\right)\right) \tag{0.14}$$

where z is the number of first nearest neighbors for a given site.

Algorithmic scheme : Translation of problem in algorithm

In this section we translate the problem described previously in an algorithm scheme. As we are modeling an evolution according to time, our code will contain an initial state and an incremental loop on time which will start from the initial time (t_0) and finish at a final time (t_{max}). During the time loop (for example between time t_n and t_{n+1}), the code will repeat the same operations which will make the matrix go from the configuration at t_n to the one at t_{n+1}. In this code we suggest that the system evolves with the following steps in the time loop:

1. Evolution of time from t_n and t_{n+1}
2. Computation of jump frequencies of all possible jumps Γ_μ
3. Drawing of a random number r_n and choice of a jump according to Eq. 0.6.
4. Completion of chosen jump: exchange of position between vacancy and nearest neighbor chosen.

Exercise

Random walks

In this first work, we model the evolution of the system if the equilibrium configuration of the alloy is an homogenized state. As we only interest ourselves to the thermodynamic evolution of the system (and not to its kinetic evolution), we assume that the alloy evolves with normalized time steps of 1 until a maximum time t_{max}. At each time step, the vacancy exchanges its position with one of its neighbors. The vacancy can exchange its position with all its first nearest neighbors X (and only its first nearest neighbors). The difference is that in this section we suppose that all exchanges have the same jump frequency Γ_{XV}. This is called a" random walk".

Preliminary work

1. Consider a vacancy located on the lattice site (xv, yv) as in Figure (0.6). In this figure, identify the first nearest neighbors of the vacancy by numbers and give the coordinates of these neighbors according to (xv, yv).

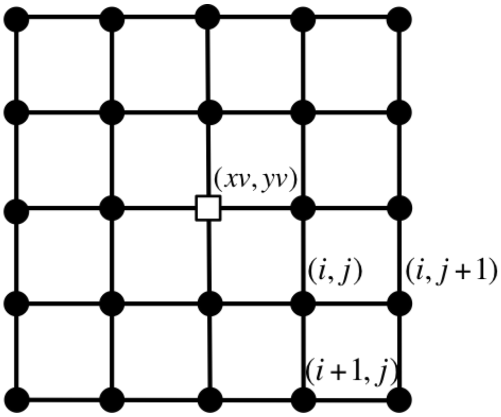

Figure 0.6: Identification of sites coordinates in the lattice and coordinates of a vacancy (represented by a square)

2. Suppose that all exchanges of the vacancy with its first nearest neighbors have the same jump frequency. Using equation (0.5), give the probability of a given jump Γ_μ.

Simulation

3. Create a folder for this MATLAB project. Open a new script in Matlab and save it in your folder as "part1.m".

4. We first write the initial state of the system in the file part1.m. Save the matrix given in the file called 'matini.mat' available from the following link:

   https://www.dropbox.com/s/8vhnam4lnburnit/matini.mat?dl=0

   Load this matrix in part1.m as "matrix". Define nx and ny as the number of rows and column respectively of matrix. In this matrix, elements A are identified by a number 1 and elements B are identified by a number 2. Place a vacancy (identified by a 0) in the middle of the matrix:

   - define in part1.m the coordinates (xv, yv) (where xv is the row and yv the column of the vacancy position) as the coordinates at the middle of the matrix.

- Place a 0 in the matrix at the appropriate coordinates (xv, yv) .

Initialize time t to 0.

5. If the matrix has the configuration of figure 0.6, what does the matrix in Matlab look like (with the numbers)? (Include a printout of the matrix).

6. Create a loop on time t where time evolves by steps of 1 as long as t remains lower than t_{max}. Place $t_{max} = 10$. We now have the part 1 in the algorithm (see section). Attach part1.m that includes all of the steps so far.

7. We now have to create the next part of the algorithm: the computation of the jump frequency of all possible jumps. (Remark: in this random walk program, this part could be placed outside of the time loop since all jumps have the same frequency. However, we include it in the time loop to prepare the second part of the problem where we will have to compute the Γ_{XV} according to the environment). In the program, we call Gamma the vector such that Gamma(i) is the jump frequency of the exchange i. Use a "for" loop to compute the values of the different Gamma(i) components.

8. We now have to choose a jump amount the different possibilities. For this, we suggest the MATLAB code shown below:

```
cumgam=cumsum(Gamma);
rn=rand;
njump=1;
for k=1:4,
    if cumgam(k)>=rn
       break
    end
    njump=k+1;
end
```

In this code, we draw a random number rn and translate the equation (0.6) in a Matlab program which uses the vector cumgam where cumgam(k) $= \sum_{i=1}^{k}$Gamma(i). According to this algorithm, if $rn = 0.3$, what is the number of the jump chosen? What is 'njump' in this code?

9. For the chosen jump, identify in your code by (xn, yn) the coordinates of the corresponding nearest neighbor according to (xv, yv). For this, we suggest you to define a matrix $(2 \times z)$ of the different possible evolutions (for example $\begin{pmatrix} +1 \\ 0 \end{pmatrix}$ or $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$) and to write (xn, yn) according to (xv, yv) and the column of the matrix corresponding to the jump.

10. We use periodic boundary conditions in this model (see part). For a site (x, y), verify that the following function enables to apply boundary conditions presented in figure (0.2)

$$x = \text{mod}(x - 1,\, nx) + 1$$
$$y = \text{mod}(y - 1,\, ny) + 1$$

For this, respond to the following questions: what is the value of x returned by this function if the x in input is between 1 and nx ? equal to 0? equal to $nx + 1$? Apply this function to xn and yn.

11. Exchange types of elements corresponding to the vacancy an the neighbor migrating in the matrix.

12. Update the vacancy coordinates to its new site.

13. In this random walk model, what is the equilibrium state of the system? (Help: the fact that all the Gamma(i) are equal induces that the migration barriers for all possible jumps ΔE_μ are equal. From equation (0.10) it induces that all saddle point interactions ε_{Ak}^{SP} and ε_{Bk}^{SP} are equal, all atom-atom interations are equal and $\varepsilon_{AV} = \varepsilon_{BV}$. What is thus the value of the ordering energy Ω in equation (0.14)? And the value of T_C ? So at any temperature, what is the equilibrium state of the system?)

14. Test: Replace the initial matrix by a matrix of same size with all A atoms on the half left side and all B elements on the half right side. Print an image of this initial matrix. Make the code run until $t_{max} = 10^6$. What do you observe? Print an image of the final matrix.

Introduction of alloy thermodynamic properties

We now have to introduce the alloy thermodynamic properties in the code. We thus have to compute the jump frequency of possible exchanges between the vacancy and its neighbors according to the alloy thermodynamic properties.

15. We recall here that $\varepsilon_{AA} = \varepsilon_{BB} = 0\,eV$. Express ε_{AB} according to the ordering energy Ω and then to the critical temperature T_C. Give a numerical value of ε_{AB} in eV if $T_C = 1000\,K$.

16. We analyze the migration barrier of an exchange between a vacancy V and one of its nearest neighbors X. We note NA the number of X first nearest neighbors of type A and NB the number of X first nearest neighbors of type B. How many first nearest neighbors does X have (we do not count the vacancy)? Express equation (0.10) according to NA, NB and $\varepsilon_{XA}$and ε_{XB} (look at the example given in section for help). Using that $\sum\limits_k \varepsilon_{Xk}^{SP} = 3\,eV$ and that $\varepsilon_{AA} = \varepsilon_{BB} = \varepsilon_{AV} = \varepsilon_{BV} = 0\,eV$, simplify the equation obtained if X is an element A. Same question if X is an element B. We observe from these calculations that, to compute the migration barrier of a jump, we need to know the type of the element of the exchange (so the type of X) and the type of all X first nearest neighbors (to compute NA and NB).

17. For a given vacancy position, we want to compute the jump frequency of the jump i (so Gamma(i)). We note X the vacancy neighbor corresponding to this jump. We start by computing NA and NB (the number of X first nearest neighbors of type A and B). We note (xn, yn) the position of X and (xnk, ynk) the coordinates of X first nearest neighbor k (k goes from 1 to 3, the vacancy position is excluded from these nearest neighbors). We write

$$\begin{pmatrix} xnk \\ ynk \end{pmatrix} = \begin{pmatrix} xn \\ yn \end{pmatrix} + nveci(k)$$

where $nveci(k)$ is the column k of the 2×3 matrix of relative position of (xnk, ynk) compared to (xn, yn). Graph 0.7 gives the position of neighbors X compared to the vacancy.

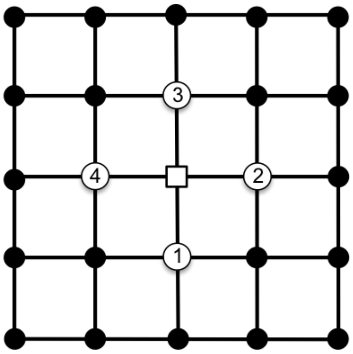

Figure 0.7: Position of X corresponding to the different possible jumps

For each of these jumps, associate the matrix nveci of the relative position of X first nearest neighbors.

$$(a)\, nveci = \begin{pmatrix} 0 & 0 & -1 \\ +1 & -1 & 0 \end{pmatrix} \quad (b)\, nveci = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$(c)\, nveci = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad (d)\, nveci = \begin{pmatrix} 0 & 0 & +1 \\ +1 & -1 & 0 \end{pmatrix}$$

18. Inside the loop to compute Gamma(i) coefficients write the following steps:

    (a) define by (xn, yn) the vacancy neighbor corresponding to i (use the nvec matrix). Apply periodic conditions to (xn, yn).

    (b) Initialize NA and NB to zero. Compute NA and NB of the exchange by analyzing the type of element on all (xnk, ynk) sites. To define the nveci matrix corresponding to the jump, you can distinguish the different cases with if-statements or you can use a structure with all the nveci matrices and load the one corresponding to the jump. Don't forget to apply boundary conditions to (xnk, ynk).

19. Express the migration barrier of each jump depending on the type of the neighbor X (located on (xn, yn)) and NA and NB. Compute the jump frequency associated to this migration barrier (place the temperature to an arbitrary value-don't forget to define ε_{AB} in your code).

20. Normalize the Gamma vector to 1 so that the sum of Gamma(i) is equal to 1.

21. Analytic calculation: Suppose that for a given position, the vacancy can exchange it's position with either of 2 different A atoms. One on them is in a local configuration with NB=0 (the jump frequency of this exchange is noted $\Gamma_{NB=0}$) and the other one is in a local configuration with NB=3 (the jump frequency of this exchange is noted $\Gamma_{NB=3}$). Compute $\Gamma_{NB=3}/\Gamma_{NB=0}$ for T=100K and for T=2000K. Explain why these ratios are consistent with the alloy phase diagram.

22. Place the temperature to 100K. Run the simulation until $t_{max} = 10^6$. What do you observe?

331 Computational Exercises

23. In binary blends where the two components have very different molar volumes (often the case with polymers), the following form of the free energy expression is commonly used:

$$\frac{v_0 G_v}{k_B T} = \frac{\phi_b ln \phi_b}{N_b} + \frac{(1-\phi_b)\ln(1-\phi_b)}{N_a} + \chi\phi_b(1-\phi_b)$$

The expressions for the chemical potentials for the A and B molecules are given by the following expression:

$$\frac{\mu_a}{k_B T} = \ln(1-\phi_b) + \phi_b(1 - N_a/N_b) + \chi N_a \phi_b^2$$

$$\frac{\mu_b}{k_B T} = \ln(\phi_b) + (1-\phi_b)(1 - N_b/N_a) + \chi N_b (1-\phi_b)^2$$

Here G_v is the free energy per volume, $\phi_b$is the volume fraction of B in the system, and N_a and $N_b$are proportional to the molecular volumes of the two components. The quantity v_0 is a reference volume, so the molecular volume of the $'A'$ molecule is $v_0 N_a$ and the molecular volume of the $'B'$ molecule is $v_0 N_b$. Adapt the MATLAB script below to do the following:

```
global na nb chi % these values also get used in function definitions
set(0,'defaultaxesfontsize',16)
set(0,'defaultlinelinewidth',2)
phi=linspace(0.001,0.999,1000);
na=150;
nb=100;
chi=0.02;
% fv is the expression for the free energy of mixing
fv=@(phi) phi.*log(phi)/nb+(1-phi).*log(1-phi)./na+chi.*phi.*(1-phi);

%mua and  mub are the chemical potentials of A and B
mua=@(phi) log(1-phi)+phi*(1-na/nb)+chi*na*phi.^2;
mub=@(phi) log(phi)+(1-phi)*(1-nb/na)+chi*nb*(1-phi)^2;

% now write the function that is equal to zero when the A and B chemical
% potentials are equal to one another for phi=phi(1) and phi=phi(2)

ftosolve=@(phi) [mua(phi(1))-mua(phi(2));
     mub(phi(1))-mub(phi(2))];

plot(phi,fv(phi));
xlabel('\phi_{b}')
ylabel('G_{v}V_{0}/RT')

% start with a guess for the equilibrium volume fractions
phiguess = [0.2; 0.8];      % Make a starting guess at the equilibrium compositions
[phicalc,fval] = fsolve(ftosolve,phiguess);   % Call solver

% now we add the tangent line
slope=(fv(phicalc(2))-fv(phicalc(1)))/(phicalc(2)-phicalc(1));
intercept=fv(phicalc(1))-slope*phicalc(1);
tangentline=intercept+slope.*phi;
hold on
plot(phi,tangentline,'r')
hold off
title(['\chi=' num2str(chi) ', N_a=' num2str(na), ', N_b=' num2str(nb)], 'fontsize',
    11)

% now save the plot as a jpg file
saveas(gcf,'commontangent.jpg');

% this saves the file a .eps file, used to embed the figure into the
% solution set - students can comment out this next line if they don't wan
% the .svg file
print(gcf,'-dsvg','commontangent.svg')
```

(a) Plot the free energy expression $\chi = 0.02$, $N_a = 200$, and $N_b = 100$, along with its common tangent construction, and report the values of the equilibrium compositions of the two phases. Include the plot with your solution.

(b) Repeat the previous calculation for $\chi = 0.025$ and $\chi = 0.05$ (you may need to adjust the initial guess for the volume fractions).

(c) Decrease χ to the critical value, χ_{crit}, which can be calculated analytically. Include a plot of G_v as a function of ϕ_b for value of χ.

24. Extend work from the previous problem to obtain a MATLAB code that generates the full phase diagram (with χ on the vertical axis and ϕ_b on the horizontal axis) for a system with $N_a = 200$ and $N_b = 100$. Show both the bimodal and spinodal curves, and turn in a printout of your actual MATLAB code.

    Hint: You'll want to solve the equations for a variety of χ values above the critical value of χ, which you can calculate analytically. You can use a 'for' loop in MATLAB to do this. Calculation of the spinodal curve is pretty easy, because there's an analytic expression for that. When you're developing your initial guesses for the bimodal compositions to pass to the 'fsolve' command, keep in mind that the bimodal compositions must lie outside the spinodal compositions. This can help you develop sensible guesses to use for this quantity.

332 Computational Exercises

MAT SCI 332 Report Guidelines

Compile the team's work into a single report. Your report should contain:

1. A brief introduction.
2. The work of each *individual* student as a separate section. You can work together on the finite element analysis, but write up the work for your component individually and combine work with labmates later.
3. A conclusion for the project as a whole.

I am looking for accuracy and deliberation in your modeling and thoughtful consideration of the results. I am not concerned with formatting or the length of your report particularly, but as for any report you write, readability, good citation sourcing, informative figure presentation, and conciseness are valued. Broadly, the body of the report should include the following:

1. The problem definition: geometry, materials properties, loads and boundary conditions.
2. Discretization: Element type and mesh parameters.

3. The results: Provide the results as a function of parameter values and discretization, as required. Show *relevant* results — displacement or stresses — and make sure to note which stress values you are showing. Note how your results change as the parameters or mesh changes.

Hint - well-constructed figures are very important to a good report. You may want to carefully design your figures so that that they are efficient at conveying relevant information.

Note that the Abaqus/CAE viewport can be exported as an image by selecting **File** ⟶ **Print**.

Submit an electronic copy of your report, as well as your Abaqus files (.cae and .odb files), by uploading them to CANVAS the date that they are due.

332 Computational Lab #1: Finite Element Analysis of a Cantilever Beam

Consider the elastic deformation of a beam. From applied mechanics, it is well-known that the cross-sectional shape of the beam has an important effect on its bending stiffness (K), or the resistance of a member to bending deformation. Here we will use the simplest example of a rectangular cross-sectioned 3D beam element to compare the Finite Element Analysis (FEA) and analytical results.

Directions: Model a cantilever beam loaded on its top surface by a *distributed* force P, as shown in Figure 0.8.

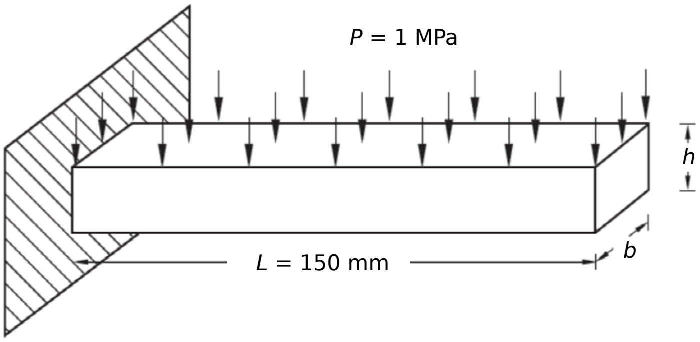

Figure 0.8: Our model for a cantilever beam.

The beam is made of a material with elastic modulus E, and a Poisson's ratio ν. Its dimensions are area $A = h \times b$, where h and b are the rectangle's height and base, respectively. The parameters we will use for this analysis are:

- $A = 500\,\text{mm}^2$
- $L = 150\,\text{mm}$
- $P = 1\,\text{MPa}$
- $E = 200\,\text{GPa}$ ($\sim$ steel)
- $\nu = 0.3$
- h and b = variable (start with 20 mm and 25 mm)

For this exercise, each group will choose 2-3 aspect ratios of $\frac{h}{b}$ (be reasonable please: not too much more than 5/1), keeping the area of the beam approximately constant, and solve for the deformation using Abaqus. For your report, you will address the following questions:

1. What is the deflection of the free end of the beam for your selected ratios? Compare to exact values from the analytical max deflection: $\delta_{max} = \frac{FL^3}{8EI}$, where F is the force, L is the length of the beam, E is the elastic modulus, and I is the area moment of inertia.
2. What are the stresses in the beam? Do your beams remain in the elastic regime under this load?
3. What is the effect of the boundary condition on the stresses in the beam?
4. What conclusions can you draw regarding the shape of the beam cross-section and the bending stiffness?

332 Computational Lab #2: Finite Element Analysis of Stress Concentrators

Stresses and strains are concentrated at holes, slots, or changes of dimension in elastic bodies. Here we will use finite elements to compute local stresses near stress concentrators. We will also utilize some new FEA techniques (non-native geometries, segmentation, seed gradients, symmetry boundary conditions, mesh density) to analyze a component that was designed in an external program and used as a source file for a 3D printed part. This computational lab is in many ways analagous to the experimental exercise completed earlier in the course.

Directions: Load and modify a CAD file (previously used for as source file in your 3D printing lab). Modify it to produce an accurate simulation of a stress concentrator (a circular or elliptical hole) as shown in Figure 0.9. As in the last lab, you can work together, but you are each individually responsible for presenting one model in your report.

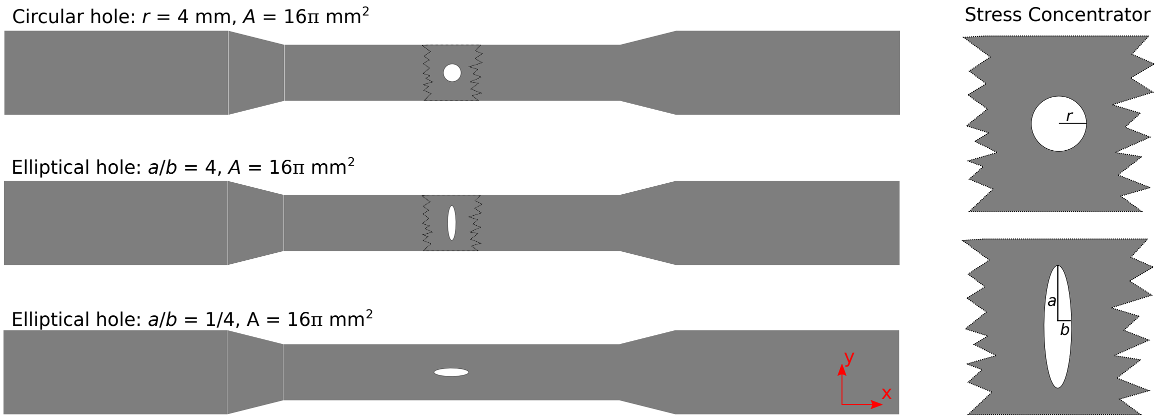
Circular hole: r = 4 mm, A = 16π mm^2
Stress Concentrator
r
Elliptical hole: a/b = 4, A = 16π mm^2
a
b
Elliptical hole: a/b = 1/4, A = 16π mm^2
y
x

Figure 0.9: Dog-bone specimens with circular (radius r) and elliptical stress concentrators (dimensions a and b).

The 3D-printed dog-bone specimen made from polymer proved to have non-isotropic tensile behavior and is too complex of a problem at this level. Instead, we will consider a dog-bone specimen of identical dimensions made of steel (elastic modulus $E = 209 \times 10^3$ MPa, and a Poisson's ratio of $\nu = 0.3$. We will use the .SAT files to define the dimensions of the material to ASTM standard. For the stress concentrators (holes) make sure that the total extruded area is equal to 16π mm^2 Use a total applied force of 10 MPa and use element type C3D8R.

The directions for this lab will be less explicit than the previous Cantilever Beam Lab. Your goal is to construct as good of a model as possible. You know how to construct an Abaqus simulation, but here are some tips for approaching this more advanced problem:

1. **Working with Non-native Geometries:** It's perfectly possible to reproduce the ASTM D638 Type I geometry using native design tools in Abaqus/CAE. However, you already have a .SAT file from your earlier work (available on Canvas in ./Computational Lab Documents/Lab #2/). Abaqus can

read this file format. Simply go to **File** → **Import-Part** and browse to find the ASTM D638 Type I.SAT file. Rename the part as you like, and accept defaults for the Part Filter. Your part should appear in your workspace.

2. **Modeling Forces/Boundary Conditions:** You need not perfectly model the serrated grips' interaction with the specimen shoulder (although you could), but carefully consider the forces on your specimen when designing your boundary conditions.

3. **Symmetry Operations:** One of the best ways to reduce computational expense is to take advantage of symmetry in your model. You can use cleverly defined symmetry conditions to reduce your mesh size by and arrive at the most elegant FEA solution.

4. **Meshing Algorithms:** The accuracy of FEA simulations are highly dependent on meshing. If possible, you want to avoid irregular meshes with elements that have high skews or low aspect ratios. The native dog-bone geometry, as loaded, requires a sweep mesh algorithm (it is yellow in the mesh module), which often works, but can lead to mesh defects. It is possible to use *partitioning* to create more regular segments that can be meshed using structural meshed algorithms.

5. **Mesh Density and Seed Gradients:** Your results will be greatly effected by your mesh density in the regions of high stress (near the concentrator). Too coarse of a mesh will yield imprecise stress values. You should systematically adapt your mesh to ensure sufficient mesh density but (somewhat) efficient computation. Use of seed gradients are an excellent way to do this (hint.)

6. **Python Scripting (totally optional):** Scripting can greatly speed your analyses. If you'd like, look at this guideby J. T. B. Overvelde to get started.

For your write-up, please make sure to address the following:

1. Describe/show how the specimens deform.

2. Describe/show the stresses in the specimen. Relate this to your theoretical understanding of stress concentrators as discussed in class.

3. How do your stresses depend on mesh density? Note, you will have to examine a number of mesh densities per specimen to investigate this.

4. What conclusions can you draw regarding the stress concentration at hole with various shapes?

5. Where do the maximum stresses occur in the specimen's long axis? σ_{11}, σ_{22}, or σ_{33}, depending on your coordinate system)? Compare the maximum value with the applied stress.

6. Prove that your mesh density near the stress concentrators is sufficient. To do this, plot the maximum short-axis σ values and long axis σ values relative to the applied stress as a function of the number of elements around the hole. Compare these relative values with the theoretical solution for the stress concentration near a circular hole in an infinite plate.

And, as always:

1. Make sure you use units in your report.

2. Label all graphs, figures, and tables appropriately (legend, axes, units, captions, etc.)

3. Cite your sources.

4. Upload the reports to Canvas as a .pdf. Include your (intuitively named) .cae and .odb files.