CS 251: Simple Science: Inlab

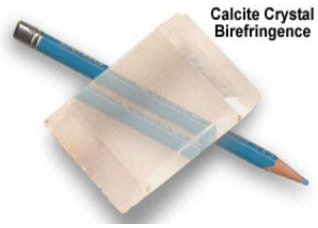
- Due: 7/26 8:10 pm Released 5:30 PM. 7/26
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on Piazza.

Overview

The goal of this assignment is to make you familiar with a scientific computing environment like Matlab/Octave. Octave is well documented at http://www.gnu.org/software/octave/octave.pdf. Another interesting tutorial for Octave can be found at http://www.dm.unibo.it/~lenci/teaching/14/maa/octavetut.pdf. You will need constructs such as `for`, `while`, `if`, `fopen` all similar to standard C. In addition, look for specialized scientific functions, but do not use third party toolboxes (packages) since we won't.

Bi-refringence

Birefringence is the optical property of a material having a refractive index that depends on the polarization and propagation direction of light. These optically anisotropic materials are said to be birefringent (or birefractive). The birefringence is often quantified as the maximum difference between refractive indices exhibited by the material but what do you care? The picture below clarifies. Crystals with non-cubic crystal structures are often birefringent.



In birefringent materials the refractive index is not a scalar in general but a *tensor*. It can be quantified with two numbers η_x and η_y which are the refractive indices parallel and transverse to the polarization axis, respectively.
A ray of light is incident on a crystal at an angle i. The polarization axis is normal to the surface. The refractive index of the crystal(η) depends on the polarization and propagation direction of light. η **varies as:**

$$\eta(i) = \eta_x \cos i + \eta_y \sin i$$

Task A1: Write an Octave function to find the incident angle i such that the incident ray is trisected after refraction, i.e.,

$$r = \frac{i}{3}$$

Input:	Read from the file: `input_inlab_task_A1.txt`. Constants: η_x and η_y (space separated values)
Output:	Write to file: **output_inlab_task_01.txt** Line 1: Value of i (error of 10^-4) is acceptable.
Submission:	Submit a file **refraction.m** and any other associated files, if any, you wrote.

Lights Out!

Lights Out is a popular mathematical puzzle (game). According to Wolfram Math World, "Lights Out is a one-person game played on a rectangular lattice of lamps which can be turned on and off. A move consists of flipping a "switch" inside one of the lamps (or squares, or buttons), thereby toggling the on/off state of this and all four vertically and horizontally adjacent squares. Starting from a randomly chosen light pattern, the aim is to turn all lamps off."

For a working demo of the game and a solver, visit this page. The page also contains the mathematical reasoning behind the solution of the puzzle which we will get into, but you should play this game now before reading further.

Algorithmic solution for 3×3

In the light chasing algorithm, you turn off lights from top to bottom by clicking buttons row by row until all rows up to the bottom are off [1]. At this point, you should consult Table 1, click buttons on the top row and repeat the process.

Bottom Row Pattern	Top Row Button to press
001	011
010	111
011	100
100	110
101	101
110	001
111	010

Table 1: Lookup Table. Left column indicates a pattern; the right, buttons.

Task B1: Write an Octave function `lookup_solve_3` which takes a single argument (a 3×3 matrix representing the initial state of the board). It returns a 3×3 matrix denoting the actions to be taken to achieve lights out. The function is to be written with the aid of Table 1.

Example

$$\text{Input} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\texttt{lookup\_solve\_3}(\text{Input}) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Submission:	Submit the file: `lookup_solve_3.m` and any other files you wrote. The function must return the appropriate value when called on the input matrix. Ignore the cosmetic long square bracket, just output the numbers.

Mathematical Solution

Lights Out (it turns out) is a memory-less game, where the current state of the board encodes all the information associated with the game. The state of the board depends on the state of each square. The state of a square depends on how often it, and its neighbors are pushed, but only whether the number of times is odd or even. Pushing a square twice is as good as not pushing it at all.

[1] This is a loaded statement, and forces you to think: what does "up to" mean? Does it exclude the bottom row or not? which row do I start with? The goal of SSL is also to improve your critical reading ability (technical) which will enable you to write better.)

Flipping each switch can be viewed as a linear transformation on the state of the board. Your background in linear algebra will tell you that such linear transformations can be encoded in a matrix describing the relation of an action to its effect on the board.

Buttons can be numbered (or as we say in CSE, indexed) in any consistent way for the following definitions. One simple way to do this would be to number the buttons on a 3×3 board as

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

We thus have,

- A, where $A_{ij} = 1 \iff$ button i is toggled on pressing button j
- b, the initial state of the board where $b_i = 1 \iff$ button i is initially on
- x, the action vector where $x_j = 1$ if button j was flipped

Given these definitions, the changes caused by actions x are represented by $Ax \mod 2$. Thus, the state of board b after actions x is $(b + Ax) \mod 2$ (since, amazingly for the purpose of this problem, mod 2 is distributive). The goal for this inlab is to find the final state of the board given the initial state and the actions taken.

Task B2: Write a function `final_state_3` in Octave which takes as input two 3×3 matrix describing the initial state and the switches pressed, and outputs a 3×3 matrix representing the final state of the board.

Example With,

$$\text{Initial} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\mathit{Action} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\texttt{final\_state\_3}(\mathit{Initial}, \mathit{Action}) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Submission:	Submit the file: `final_state_3.m` and any other files that you might have created. The function must return the appropriate value when called on the input matrices.

How We will Grade You

The number of points per task (A1, B1, B2) appears below

1. Relevant comments in the code, honor code, citations, brief reflection essay: 20 points.
2. Correct answer: 80 points

Submission Guidelines

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10%. In this example, the second student will get full marks (10/10) and the first student will receive 8/10.
2. Do include a readme.txt (telling me whatever you want to tell me). Do include group members (name, roll number), group number honour code, citations etc.

3. The folder and its compressed version should both be named `lab01_groupXY_final` for example folder should be named `lab01_group07_final` and the related `tar.gz` should be named `lab01_group07_final.tar.gz`