CS1132 Spring 2016 Assignment 1b Due Mar 9th

Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 4 of this document.

Do not use the `break` statement in any homework or test in CS1132.

1 Natural logarithm approximation

The natural logarithm of $(1 + x)$ can be approximated using Taylor series when $x \in (0, 1)$ (Equation 1). When we use a finite number of terms in the Taylor series, the error incurred in the approximation is called the *residual*, denoted by $R_n(x)$; the residual can be bounded by Equation 2 when we use n terms. In this assignment you are going to approximate the natural logarithm and analyze the residual.

$$ln(1+x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} x^n = x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots + (-1)^{n+1}\frac{x^n}{n} + \cdots \qquad x \in (0,1) \tag{1}$$

$$|R_n(x)| \leq \frac{x^{n+1}}{n+1} \qquad x \in (0,1) \tag{2}$$

1.1 My log function

Write a function **myLn.m** as specified below. Use a while-loop to repeatedly add terms in the Taylor series until the residual is strictly below the threshold.

```
function [result, n] = myLn(x, threshold)
% Estimate ln(1+x) using Taylor series.
% x is a number in (0, 1).
% result is the estimated ln(1+x).
% n is the smallest number of terms used in the Taylor series such that the
% error of the approximation is strictly below threshold.
```

1.2 Residual analysis

Your second task is to write a script **plotResidual.m** that draws the two figures specified below. In this script you do **NOT** need to use **myLn.m**.

In the first figure, plot three curves that represent the upper bound of the residual vs. the number of terms n (Equation 2), for $x = 0.2, 0.5$, and 0.8 respectively. The x axis represents the number of terms, which should be from 1 to 25. The y axis represents the bound of the residual. Use different colors for the three curves and use `legend` to indicate the corresponding x value for each curve. Use `xlabel`, `ylabel` and `title` to add meaningful labels and title to the figure.

In the second figure, plot a curve that represents the upper bound of the residual vs. x, when approximating $ln(x + 1)$ using $n = 5$ terms. The x axis represents the value of x, which should be `0.01:0.01:0.99`. The y axis represents the bound of the residual. Use `xlabel`, `ylabel` and `title` to add meaningful labels and title to the figure.

2 Reversi game

Reversi is a board game for two players, played on an 8×8 board. There are 64 identical disks that can be placed on the board, each of which has a white side and a black side. The two players are assigned the white and black color respectively, and take turns to place disks on the board with their assigned color facing up. When the current player places a new disk on the board, any disks of the other player's color that are in a straight line, and which is bounded by the new disk and another disk of the current player's color, need to be reversed to the current player's color. When there is no empty square on the board where a new disk can be legally placed, the player who has the majority of disks displaying his/her color wins the game. You can find online Reversi games and play it to better understand the rules. In this assignment, you are going to simulate this game between two AI players.

In this assignment, we represent the board with an 8-by-8 matrix. Each entry of the matrix can have three values: -1 means that there is no disk at this position; 0 represents a black disk and 1 represents a white disk. We will provide you the scripts for drawing the board and simulating the game. Once you have finished the two scripts **placeNewDisk.m** and **pickNewPosition.m**, you should be able to run **reversiSim.m** and see an animation of the game.

2.1 Placing a new disk

When a player attempts to place a new disk, only a few positions are *legal*, based on the current state of the board. A legal position must have these two properties:

- There is no disk at this position.
- At least one disk of the opponent's color will be reversed after placing the new disk at this position.

After placing a new disk of color A, some disks of the opposite color B on the board must be reversed: if a straight line of color B disks is bounded by the new disk and another color A disk and there is no gap between these disks, then those color B disks should be reversed. Figure 1 shows a few steps in which disks are placed and reversed.

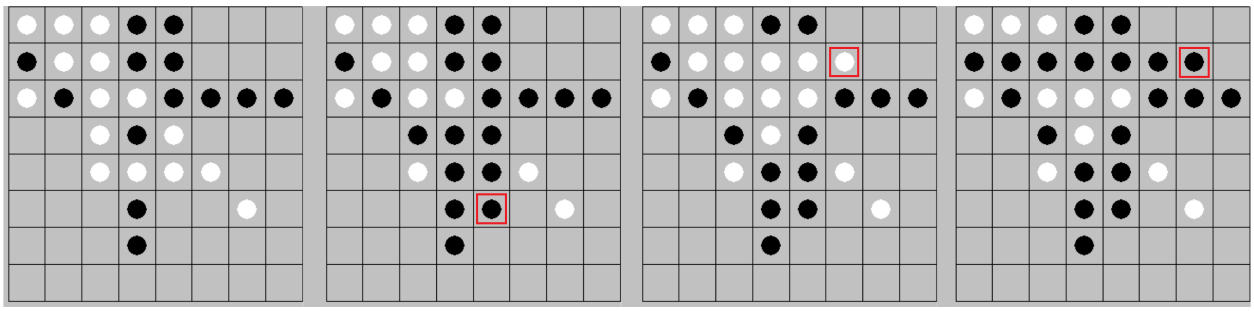

Figure 1: States of the board. After placing a new disk (red squares), some disks of the opposite color need to be reversed.

Your job is to implement the following function that tries placing a new disk at a given position and update the state of the board if it is a legal position, as specified.

```
function [newBoard, numReversed] = placeNewDisk(board, r, c, color)
% Try placing a new disk at a given position
% board is an n-by-n matrix representing the state of the board before
% placing the disk.
% Attempt to place the new disk at board(r, c). r is the row number and c
% is the column number. In drawBoard(), r increases going south and c
```

```
% increases going east.
% color = 0 when we try to place a black disk. color = 1 for a white disk.
% newBoard is the state of the board after placing the new disk.
% numReversed is the number of disks reversed after placing the new disk.
% If (r, c) is not a legal position, newBoard = board and numReversed = 0.
```

HINT: You need to check for eight half-lines shooting from the position of the new disk. Write a subfunction that counts the number of reversed disks along one half-line in a given direction, and use this subfunction to avoid redundant code. In this subfunction, you can use a while loop to do the counting. Starting from the position next to the new disk's position, take one step forward in each iteration, until you reach a position that is not occupied by a disk of the opposite color or go out of bound. After exiting the loop, if you are at a position where there is a disk of the current player's color, then you need to reverse the disks along the way. Otherwise no disk needs to be reversed. An example of this subfunction is specified below.

```
function numReversed = countReversed(board, r, c, color, xi, yi)
% Count the number of reversed disks along a particular direction after
% placing a new disk.
% board, r, c, color have the same meaning as in placeNewDisk.
% xi and yi indicate the direction. e.g. xi=1 and yi=1 means that we are
% counting along the half-line in the southeast direction.
% numReversed is the number of reversed disks along this direction.
```

2.2 Picking a position

Given the current state of the board, the AI should be able to decide on a position to place a new disk. Developing sophisticated AI for the game is beyond the scope of this course and you are asked to use a simple greedy strategy. That is, place the disk at a position such that the maximum number of disks will be reversed after placing it there.

Implement the function **pickNewPosition** as specified below. In this function, you should sweep the matrix **row by row from top left to bottom right**. For each entry of the matrix, use **placeNewDisk** to determine how many disks will be reversed if the new disk is placed at that position. Return the position that maximizes the number of reversed disks. If multiple positions result in the same maximum number of reversed disks, return the position that occurs first during the sweeping.

```
function [r, c] = pickNewPosition(board, color)
% This function finds a legal position to place a new disk such that the
% number of reversed disks is maximized. If multiple positions have the
% same maximum number, pick the one that occurs first when sweeping the matrix.
% board is the state of the board before placing the disk.
% color is 0 for black disk and 1 for white disk.
% The new disk is placed at board(r, c).
% If there is no legal position, r = 0 and c = 0.
```

3 Self-check list

The following is a list of the minimum *necessary* criteria that your assignment must meet in order to be considered *satisfactory*. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time.
Note that, although all of these are necessary, meeting all of them might still not be *sufficient* to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Δ Comment your code! If any of your functions is not properly commented, regarding function purpose and input/output arguments, you will be asked to resubmit.
- Δ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Δ Make sure your functions names are *exactly* the ones we have specified, *including* case.
- Δ Check that the number and order of input and output arguments for each of the functions matches exactly the specifications we have given.
- Δ Test Each one of your functions independently, whenever possible, or write short scripts to test them.
- Δ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check this by running each script several times in a row. Before each test run, you should type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

4 Submission instructions

1. Upload files `myLn.m`, `plotResidual.m`, `placeNewDisk.m` and `pickNewPosition.m` to CMS in the submission area corresponding to Assignment 1b in CMS.
2. Please do not make another submission until you have received and read the grader's comments.
3. Wait for the grader's comments and be patient.
4. Read the grader's comments carefully and think for a while.
5. If you need to resubmit, fix all the problems and go back to Step 1! Otherwise you are done with this assignment. Well done!