CS1112-206/211/212–Fall 2009

Lab 10 Solution

Friday, November 6

There are many ways to deal with cell arrays and vectors, because we have a choice to use vectorized or non-vectorized code. Some times non-vectorized code is more understandable than vectorized code. Other times the reverse is true. This solution contains vectorized code if the solution on the course webpage uses non-vectorized code, and vice versa.

1 Cell Array vs Vector

The commands on the worksheet are instructive with comments on the side. No further explanation will be provided here. Be sure to use braces (`{}`) for cell array and parentheses for vectors. For vectorized code, it is necessary to use parentheses even if the variable is a cell array.

2 Deck of Cards

The function `DispCards` is straightforward and will not be discussed further. Be sure to use braces to access each component of a cell array.

For `MyShuffle`, what we have to do is the following:

- Pick a random point to cut the card.
- Cut the cards.
- Alternate the cards until one deck is empty.
- Put the leftover deck to the end of the shuffled deck we are building.

Since the code posted on the course webpage is non-vectorized, we present a vectorized solution. Note that since alternating process is more understandable when the code is non-vectorized, we leave it non-vectorized in our solution. See `MyShuffle.m`. The vectorized version, if you are interested, is presented below:

```
sd(1:2:2*numAlt-1)=TopD(1:numAlt);
sd(2:2:2*numAlt)=BotD(1:numAlt);
```

3 Structure and Structure Array

The function `MakeSquare` is straightforward and will not be discussed further. Now we need to write a script that creates an array of structures. It turns out that there is no easy, crystal clear way to preallocate space for structure (as opposed to `zeros` and `ones` for vectors, `blank` for strings (one-dimensional), and `cell` for cell arrays). We just have to be inefficient and grow the array inside the loop. Just a reminder that the dot (`.`) operator provides a way to access a component in a struct.

4 More Card Playing...

The solution posted on the course webpage is non-vectorized code. The solution below is (totally) vectorized.

```
function sd=Cut3(d)
% d is a one-dimensional cell array of strings whose length is a multiple of 4.
% sd is the cell array after cutting the deck by taking half the cards from
%   the middle of the deck and putting that half on top.

num=length(d);
numPart=num/4;

sd=[d(numPart+1:3*numPart) d(1:numPart) d(3*numPart+1:num)];
```