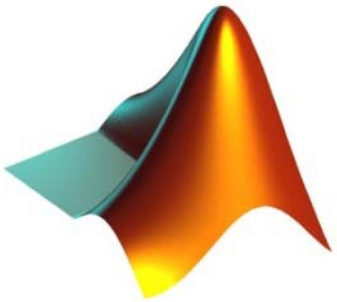


Matlab Review

Lecture 14 (Mar 9)
CS100M – Spring 2006

Announcements

- Project 3
  - Due today
- Prelim II
  - Thursday, March 16, 7:30pm
  - You *must* contact Kelly Patwell (see website) if you have any scheduling difficulties
  - Room assignments: announced next week and on the Web
  - Prelim 2 topics: Everything through today
    - Material introduced next week will not appear on the prelim
  - Review session
    - This Sunday (see website)
    - Review problems will be online soon

Topics

- Reading: No new reading
  - We have read online Chapters 1, 2, 3, 4, 5, and 9
- Recall recent topics
  - 1-dimensional arrays (vectors)
  - 2-dimensional arrays (matrices)
  - Characters and strings
  - Simulation and use of random number generator
  - Vectorized code
  - Simple plotting
  - Logical arrays

Neighborhood of a Cell

- We define the *neighborhood of a cell* to be the cell itself and all adjacent cells (including diagonally adjacent)

7	0	7	0	5
2	4	5	2	6
4	6	3	8	1
7	0	5	2	4
3	8	6	2	1

The neighborhood of cell(2,4)

The neighborhood of cell(5,2)

Min of a Neighborhood

- Goal:
  Write a function minInNeighborhood(M, row, col) that reports the minimum value in neighborhood of cell(row, col) in matrix M
- Function header

```
Function val = minInNeighborhood(M, row, col)
% Return min in neighborhood of (row, col) in M
```

Ask Yourself Questions

- Do we know how to solve a similar problem?
  - Yes, we already have code to find the min of a matrix
- Can we make a neighborhood into a matrix?
  - Yes, Matlab makes it easy to do submatrices
  - Neighborhood of M(row, col) is M(row-1:row+1, col-1:col+1)
- What happens near the edges?
  - Doesn't work near the edges: we "fall off"
- What can we do to fix up the edges?
  - We can make the code more complicated, or…
  - We can modify the matrix so we *can't* fall off
- If we add a border around M, what goes in the border?
  - realmax

Example: Random Walk

- Write a function randomWalk(n) to perform n steps of a random walk in the plane starting from (0,0)
  - Function header: function randomWalk(n)
- At each step, possible moves are up, down, left, or right
- Display the walk
  - This part turns out to be easy
  - plot(x, y, '-') where x and y are vectors draws connecting lines from (x(0), y(0)) to (x(1), y(1)) to (x(2), y(2)) to…

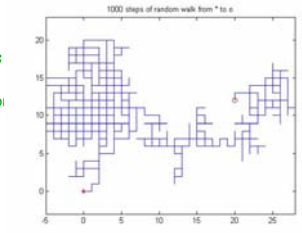
Ask Yourself Questions

- How do we know what to do at each step?
  - We use rand( ): there are 4 equally likely directions
- How can we draw the random path?
  - Plot( ) makes this easy
  - We need to know all the x-values and all the y-values
  - Note: It's easier to draw the entire path than to draw one piece at a time
- How do we store the random path?
  - We can use a single n-by-2 matrix, or
  - We can use an n-vector of x-values and a separate n-vector of y-values
- Does this make sense for one step?
  - No, for one step we need…
    - The starting position (0,0)
    - And one step to either (1,0), (0,1), (-1,0), or (0,-1)
  - Thus, we should be using n+1 instead of n

Random Walk Algorithm

- Pseudocode

```
Load x and y with n+1 zeros
for each step k
   Choose a random directio
   Update x(k+1) and y(k+1)
Draw the result
```



Vectorized-Code Examples

- Write code to reverse a string
  - s = s(end:-1:1);
- Write code to modify an integer matrix so that all even values are set to 4 and all odd values are set to 3
  - L = (mod(A,2) == 0);
  - A(L) = 4; A(~L) = 3;
- Write code to produce a random sequence of H's and T's (for Heads and Tails)
  - L = (rand(1, 50) < 0.5);
  - s(L) = 'H';
  - s(~L) = 'T';
- Write code to "rotate" a matrix clockwise
  - B = A';
  - A = B(:, end:-1:1);

Recall: Capitalize First Letters

- We did this before with iteration (i.e., loops)
- Can use vectorized code instead
  - It's not clear that this is better
- Idea: Everything after a blank should be capitalized

```
L = (s == ' ');          % Find all the blanks
L = [ true  L(1:end-1) ] % Shift each blank to right
S = upper(s);            % This capitalizes everything
s(L) = S(L);             % Copies just parts of S into s
```

Plotting Examples

- Plot two cycles of the sine function

```
x = 4 * pi * (0:.01:1);   % Choose 100 x-values
y = sin(x);               % Find sine for each x
plot(x, y);               % Plot sin(x) using default colors
```

- Plot two cycles of the cosine function on the same graph

```
z = cos(x);               % Find cosine of each x
plot(x, y, x, z);         % Plot both sin(x) and cos(x)
```

- Same, but use dotted lines

```
plot(x, y, ':', x, z, ':')
```