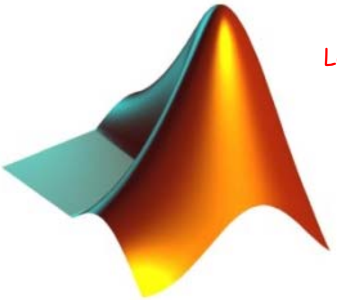


Last Matlab Lecture

Lecture 13 (Mar 8)
CS100M – Spring 2007

Announcements

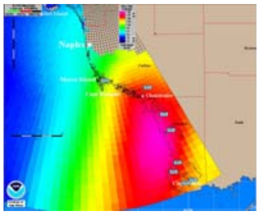
- Prelim II
  - 7:30pm, Thursday, March 15
  - If you have an exam conflict
    - Contact Kelly Patwell (Course Administrator) ASAP
  - Prelim 2 topics: Everything through today
    - Material introduced next week will not appear on the prelim
  - Review session
    - This Sunday (see website)
    - Review problems are online
- This is the last CIS/EAS 121 lecture

Topics

- Reading: CFile 9, Section 9.3
  - 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
  - Vectorized code
  - Simple plotting
- Today
  - Simulation using the random number generator
  - Logical arrays

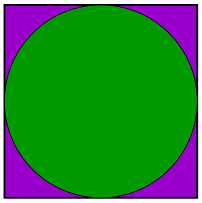
Simulation

- The application of mathematical and computer models to imitate the behavior of a system
  - Usually a real-world system (but not always)
  - Useful for design, training, & games
- Matlab provides many tools useful for simulation
  - We'll examine some very simple simulations



Example: Simulation of Darts

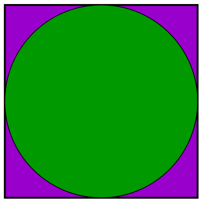
- Goal: Simulate darts thrown at a simple target to derive an estimate of π
- We did this example earlier using iteration
- Assume hits are distributed uniformly over this 2-by-2 square
  - $N_{in}/N = A_{circle}/A_{square} = \pi/4$



Original Code (for Just One Throw)

```
close all
hold on
axis('equal');
axis([-1 1 -1 1]);

px = 2*rand - 1;
py = 2*rand - 1;
if (px^2 + py^2 <= 1)
    plot(px, py, 'og');
else
    plot(px, py, 'or');
end
```



Throwing Darts using Vectorized Code

- How can we compute all throws at once by using a nDarts-by-2 matrix?
- How can we determine each throw's distance from origin?
- How can we count how many of the throws are within the circle?

```
function estimate = approxPi(nDarts)

throws = -1 + 2*rand(nDarts, 2);
x = throws(:, 1);
y = throws(:, 2);

dist = sqrt(x.^2 + y.^2);
in = sum(dist <= 1);
estimate = 4 * in/nDarts;
```

Example: Rolling a Fair Die

- Goal: Simulate the rolling of a fair die and create a histogram of the outcome
- How can we compute all the die rolls at once?
- How can we count how many of each roll occurred?

```
function count = rollDie (nRolls)

count= zeros(1,6);
rolls = ceil(6 * rand(1, nRolls));

for k= 1:6
   count(k) = sum(rolls == k);
end
```

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…

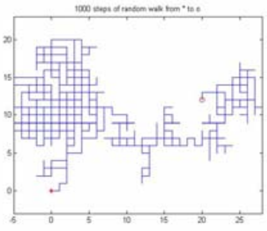
Ask Yourself Questions

- 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
```



Logical Subscripts

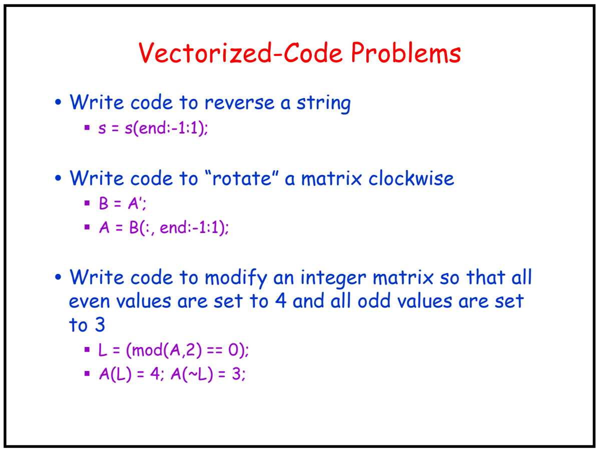
- Recall logical arrays
  - Occur when you use vectorized relational operators
  - Consist of 0's (for false) and 1's (for true)
  - The Workspace viewer (in the Desktop menu) shows the "class" of each of your variables
- Logical arrays can be used as subscripts!
  - The shapes must match
- Examples

```
M = [7 0 5; 2 4 6; 3 8 1]
M(M<4) = 99
```

  - All values < 4 are set to 99

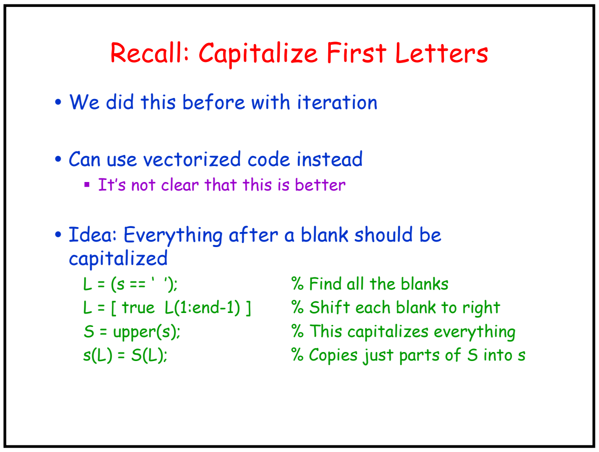
```
s = 'this is a string'
s(s<'n') = 'X'
```

  - All letters in the first half of the alphabet are replaced with 'X'



Vectorized-Code Problems

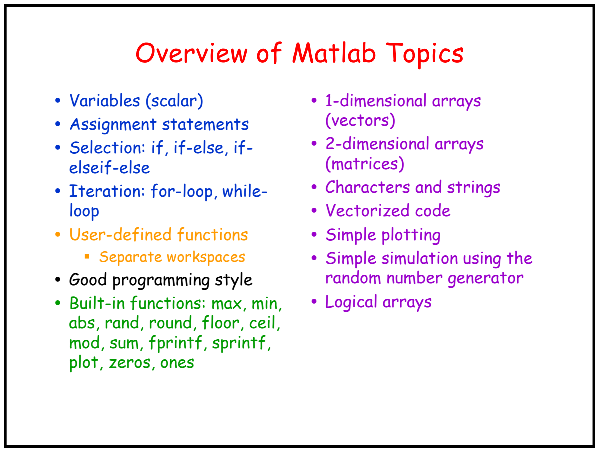
- Write code to reverse a string
  - s = s(end:-1:1);
- Write code to “rotate” a matrix clockwise
  - B = A’;
  - A = B(:, end:-1:1);
- Write code to modify an integer matrix so that all even values are set to 4 and all odd values are set to 3
  - L = (mod(A,2) == 0);
  - A(L) = 4; A(~L) = 3;



Recall: Capitalize First Letters

- We did this before with iteration
- 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
```



Overview of Matlab Topics

- Variables (scalar)
- Assignment statements
- Selection: if, if-else, if-elseif-else
- Iteration: for-loop, while-loop
- User-defined functions
  - Separate workspaces
- Good programming style
- Built-in functions: max, min, abs, rand, round, floor, ceil, mod, sum, fprintf, sprintf, plot, zeros, ones
- 1-dimensional arrays (vectors)
- 2-dimensional arrays (matrices)
- Characters and strings
- Vectorized code
- Simple plotting
- Simple simulation using the random number generator
- Logical arrays