CS 100M: Lecture 8
February 17
P2 Due Today @ 6pm
Prelim 1 on Thursday Feb 24

Prelim 1 Syllabus
Assignment, if, if-else, if-elseif, boolean expressions, for-loops, while-loops, built-in functions mod, floor, ceil, round, sin, cos, exp, abs, log, log2, rand, etc.
Look at P1 and P2 solutions.
Look at all lecture examples through today.
Look at all section and lab examples.

Review Session
Sunday 1:00-2:30pm
Location: Check Website
Also, check website for sample exam questions that will be covered at the review session

User-Defined Functions
You can define your own functions in Matlab.
One is motivated to do this for the same reasons as in mathematics:
- Encapsulates key (computational) ideas
- Facilitates reasoning
- Permits a division of labor

Structure for the Simplest Situation

```matlab
function < output var > = < name>(< input var >)
% Post: < Properties of the output variable >
% Pre: < Restrictions on the input variable >

Function Body
```

Writing / Designing Functions
Will do two examples.
How do we identify something useful to encapsulate as a function?
Eg 1: First to N

- Two players H and T toss a coin
- H scores a point if outcome is heads
- T scores a point if outcome is tails
- First to score N points wins

Question: What is the average number of tosses required for a game for N = 5, 10, 15, ..., 100

The Top Level Script...

N = 5;
G = large number;
while N <= 100
    Play First-to-N G times and compute average.
    Display the average.
    N = N + 5;
end

Think: function

N = 5;
G = large number;
while N <= 100
    m = AveToss(N,G)
    Display the average m
    N = N + 5;
end

Specify AveToss

function m = AveToss(N,G)
% Post:
% m is the average number of tosses
% in a game of First-to-N based on
% a sample of G games
% Pre:
% N and G are positive integers

Developing AveToss

function m = AveToss(N,G)
    sum = 0;
    for i = 1:G
        Play a game of First-to-N
        sum = sum + H’s score + T’s score
    end
    m = sum/G

More AveToss Development

function m = AveToss(N,G)
    sum = 0;
    for i = 1:G
        [nTails,nHeads] = FirstTo(N)
        sum = sum + nTails + nHeads
    end
    m = sum/G
Specify FirstTo(N)

function [nHeads,nTails] = FirstTo(N)
% Post
% Simulates a game of First To N.
% nHeads is the final score of player H.
% nTails is the final score of player T.
% Pre
% N is a positive integer

Observe: A function can have more than one output argument!

Developing FirstTo

function [nHeads,nTails] = FirstTo(N)

nHeads = 0; nTails = 0;
while (nHeads < N) && (nTails < N)
    x = rand;
    if x<=.5
        nHeads = nHeads + 1;  % Heads
    else
        nTails = nTails + 1;  % Tails
    end
end

Top-down problem solving

By writing clear specifications, can “work with a function” before it is implemented, i.e., before it is completely written.

E.g., Can write AveToss completely before FirstTo is finished

Eg. 2: The Newton Fractal

For any complex z₀ that is not on the 45 degree lines in the complex plane, the sequence z₀,z₁,z₂,...

\[ z_{n+1} = \frac{3z_n^4 + 1}{4z_n^3} \]

cconverges to either 1, i, -1, or -i.

Examine Convergence

Enter a starting z and see how long it takes to “get close” to one of the four “attractors” 1, i, -1, -i.

A while loop setting: don’t know how many steps it’ll take.
Write a Function to Explore This

function [zFinal,its] = NewtonFractal(z)
% Post:
%   zFinal: the attractor for z (either 1, i, -1, or -i)
%   its: the number of iterations required to get close to the attractor
% Pre:
%   z: a complex number with abs(real(z)) not equal to abs(imag(z))

A Function w/o Output Arguments

function DisplayDotProps(x,y,zFinal,m)
% Post:
% Displays x+iy in the current plot window, color-coded according to the associated attractor. The value of m is also displayed.
% Pre:
%   x+iy is the starting point for the fractal sequence and zFinal is the attractor (1, i, -1, or -i) to which the sequence is converging.
%   m is the number of steps required for the sequence values to get close to the attractor.