Repetition via Looping

Topics:

- for loops
- while loops

Learning Outcomes:

1. Understand when and how to apply loops
2. Know which looping structure is appropriate to use
3. Solve problems requiring the use of loops

Book Sections: 4.1 - 4.5

1. Background

Computers are built for repetition. Loops are the primary mechanism to perform repetive tasks and to control the flow of the program.

There are two kinds of loops, **for** and **while**. For loops are typically used when the number of repetitions (or iterations) is know, whereas *while loops* are made for when this information is unknown. For example, a for loop may be more appropriate for summing the first 100 integers, but the while loop fits the situation in which a random walk on a grid is to occur (i.e., the number of steps is unknown, therefore continue until the end is reached).

2. For loop

```
%/ Print 'I am happy' 10 times \%


for i=1:10
    disp('I am happy');
end
```

```
%/ Print elements of a list \%


% Generate random list of 20 elements
```

```
list=rand(1,20);


for i=1:20
    disp(list(i));  % prints the ith element
end
```

```
%/ Sum the squares of the first 5 positve integers \%


% Initialize variable
theSum=0;

% Add each term one at a time
for i=1:5
    theSum=theSum+i^2;
end
```

2.1 - Nested Loops

Often it is necessary to repeat a subtask for each task. For example,

```
%/ Find the sum of all elements in 2d-array \%


% Initialize variables
theSum=0;
array=randi(10,5,4);

% Iterate over each row per column
for c=1:4
    for r=1:5
        theSum=theSum+array(r,c);
    end
end
```

3. While loops

Although while loops can be made equivalent to a for loop, typically they are used when the number of iterations (e.g., the size of array) is

unknown.

Here are the for loop examples above translated into while loops.

```
%/ Print 'I am happy' 10 times \%

i=1;

while i<11
    disp('I am happy');
    i=i+1;
end
```

```
%/ Print elements of a list \%


% Generate random list of 20 elements
list=rand(1,20);

% Initialize index variable
i=1;

% Loop over list
while i<=20
    disp(list(i));  % prints the ith element
    i=i+1;
end
```

```
%/ Sum the squares of the first 5 positve integers \%


% Initialize variable
theSum=0;
i=0;

% Loop over 1:5 and add each term, one at a time
while i<5
    theSum=theSum+i^2;
    i=i+1;
```

```
end
```

3.1 Unknown iterations

While loops are most effective when the number of iterations is unknown.

3.1.1 Example: Random Walk

Consider a simple random walk in which at each step exactly one coordinate changes according to the uniform distribution. Random walks have applications in economics, population genetics, physics, computer science, mathematics, and psychology. Often it is used as a substitute for Brownian motion and also used for simulations. Random walks have been used to sample online social networks graph, used to suggest who to follow on Twitter, and estimate the size of the Web.

```
% Initialize the grid size
n=6;

% Initialize the walk counter and current location
k = 0;
xk = 0;
yk = 0;


% Start Visualization Sequence
pause on
plot(xk,yk,'r*','MarkerSize',16);
axis([-n n -n n])
pause(0.5)


% The random walk, continue until end reached
while abs(xk)<n && abs(yk)< n

    % Simulate a single step (N,S,E,W)
    r = rand(1);

    % Move one direction
    if r <= 0.25
        yk = yk+1;      % Move North
    elseif r <= 0.50
```

```
        xk = xk+1;      % Move East
    elseif r <= 0.75
        yk = yk-1;      % Move South
    else
        xk = xk-1;      % Move West
    end

    % Save the new location
    k = k+1;
    x(k) = xk;
    y(k) = yk;

    % Plot new point
    plot(xk,yk,'rs','MarkerSize',16);
    axis([-n n -n n])   % freezes the frame
    pause(0.5)          % animation effect
end

pause off
```

```
% Alternative

% Replace the if-then-else above with a switch statement

r=randi(4);

switch r
    case 1
        yk = yk+1;      % Move North
    case 2
        xk = xk+1;      % Move East
    case 3
        yk = yk-1;      % Move South
    case 4
        xk = xk-1;      % Move West
end
```

3.2 Average Length of walk

Calculate the average walk length for variable walk sizes.

```
% Calculate the avg. walk length for variable walk sizes

m=10000;    % number of simulations
n=5;        % grid size

s=0;        % sum of walks
for i=1:m
    [x,~]=randomwalks(n);
    s=s+numel(x);
end

fprintf('The average walk on grid size %i is %5.2f\n',n,s/m);
```

```
The average walk on grid size 5 is 29.11
```

```
function [x,y]=randomwalks(n)

    % Initialize the walk counter and current location
    k = 0;
    xk = 0;
    yk = 0;


    % The random walk, continue until end reached
    while abs(xk)<n && abs(yk)< n

        % Simulate a single step (N,S,E,W)
        r = rand(1);

        % Move one direction
        if r <= 0.25
            yk = yk+1;      % Move North
        elseif r <= 0.50
            xk = xk+1;      % Move East
        elseif r <= 0.75
            yk = yk-1;      % Move South
```

```
        else
            xk = xk-1;      % Move West
        end

        % Save the new location
        k = k+1;
        x(k) = xk;
        y(k) = yk;
    end

end
```

Return to MAT225