Loops

Loops

- loops allow you to repeatedly execute blocks of code
  - each repetition is called an *iteration*

- MATLAB has two kinds of loops
  - **`for`** loop
    - repeats a block of code a specific number of times, keeping track of each iteration using an incrementing loop variable
  - **`while`** loop
    - repeats a block of code as long as a logical condition remains true

for loop

- a **`for`** loop repeats a block of code once for each element of a control vector*
  - *the vector can be an array, but ignore this for now

- the value of the element in the control vector is available inside the loop

for loop

index is the loop variable; you can use whatever name you want

```
for index = vector_of_values
```

each iteration of the loop, the value of index is taken sequentially from ***vector_of_values***

loop body: a sequence of MATLAB statements repeated once for each element in ***vector_of_values***

```
end
```

```
% display the value of index for each iteration
% of a loop

for index = 1:5
  index
end
```

```
% display the integers 1 through 5 on separate lines

for index = 1:5
  disp(num2str(index));
end
```

```
% you could also use sprintf

for index = 1:5
  disp(sprintf('%d', index));
end
```

```
% display the integers start through stop
% on separate lines

if start <= stop
  x = start:stop;
else
  x = start:-1:stop;
end

for index = x
  disp(num2str(index));
end
```

```
% compute the sum of the integers 1, 2, 3, ..., n

% we need a variable to accumulate the sum
total = 0;

for x = 1:n
  total = total + x;
end


% you should use sum instead, though
total = sum(1:n);
```

```
% compute the dot product of two vectors x and y

len = length(x);
dotprod = 0;
for index = 1:len
  dotprod = dotprod + x(index) * y(index);
end


% you should use dot instead, though
dotprod = dot(x, y);
```

for loop: matrix-vector multiplication

- use a for loop to compute the product of a matrix and a vector
  - use the function **`dot`** inside of the loop

for loop: radioactive decay

- in radioactive decay, an energetically unstable atom spontaneously emits energy in the form of ionizing radiation
- for a single atom, the decay occurs at random
  - for many atoms, the decay occurs at an average constant rate
- suppose that you start with N radioactive atoms:
- after 1 unit of time there will be:

for loop: radioactive decay

- suppose that you start with N radioactive atoms:

$$U(1) = N$$

- after 1 unit of time there will be:

$$U(2) = (1 - \alpha)U(1)$$

for some constant value α

- after 2 units of time there will be:

$$U(3) = (1 - \alpha)U(2)$$

- and so on

```
% compute the number of atoms at each time
% t = 1, 2, 3, ..., 10

N = 100000;
alpha = 0.05;

% we need a vector to store the results
U = zeros(1, 10);
U(1) = N;

for index = 2:length(U)
  U(index) = (1 – alpha) * U(index – 1);
end

% what is the better way to compute U?
```

for loop: cumulative sum

- the cumulative sum of the elements in a vector **`values`** is a vector of the same length as **`values`** where the element at index **`i`** is the sum of **`values(1)`** through **`values(i)`**

```
% compute the cumulative sum of the elements
% in a vector named values

csum = zeros(1, length(values));
csum(1) = values(1);
for index = 2:length(values)
  csum(index) = csum(index - 1) + values(index);
end


% you should use cumsum instead, though
csum = cumsum(values);
```

while loop

- a **`while`** loop repeats a block of code as long as a logical condition is true
  - unlike a for loop
    - there is no loop variable
    - the number of times that the loop runs is not necessarily determined ahead of time

while loop

while *logical_condition*

loop body: a sequence of MATLAB statements

end

if logical_condition is true then the loop body is run once

after the loop body is run, the loop restarts by checking the logical_condition

```
% repeat a loop until the user inputs 'y'

repeat = 1;
while (repeat)
  %
  % some code here that you want to repeat
  %

  % ask the user if they want to repeat again
  answer = input('Continue? (y / n)');
  repeat = strcmp(answer, 'y');
end
```

while loop: infinte loops

- observe that it is very easy to create an infinite loop using a `while` loop
  - you must ensure that whatever happens in the loop body eventually causes the logical condition to become false

- if you encounter an infinite loop in your program you can press `Ctrl + c` to stop your program
  - unfortunately this stops your entire program and not just your loop

```
% infinite loop example

repeat = 1;
while (repeat)
  %
  % some code here that you want to repeat
  %

  % ask the user if they want to repeat again
  answer = input('Continue? (y / n)');

  % comment out next line
  % repeat = strcmp(answer, 'y');
end
```

while loop: computing square root

- Heron's method
  - named after Hero of Alexandria (1st century Greek mathematician)
- to compute the square root of s

1. choose a starting value x_0
2. let x_1 be the average of x_0 and s/x_0
3. let x_2 be the average of x_1 and s/x_1
4. let x_3 be the average of x_2 and s/x_2, and so on

- how do you know when to stop?

```
% compute the square root of s

epsilon = 1e-9;
delta = Inf;
x = 0.5 * s;
while abs(delta) > epsilon
  xi = mean([x, s / x]);
  delta = xi – x;
  x = xi;
end
```