The purpose of a loop is to execute the same commands or calculations multiple times without having to explicitly type it out every single time. Each pass through the commands or calculations is called an **iteration**. There are two types of loops in MATLAB: while loops and for loops.

A **for loop** is used for a **set number of iterations**, normally determined by the size of your data set. The for loop is useful when you have a domain you want to operate in.

|  |  |  |
| --- | --- | --- |
| for j = <1D vector>  <executable>;  end | j is called an **indexing variable**. On each iteration, j takes on the value of the next value in the vector you have specified. Examples 1-3a are all valid ways of setting the indexing variables.[[1]](#footnote-0) | |
| **How many iterations will each of the following examples go through? What value does j take on in the fourth iteration in each example?** | | |
| **Ex 1a.**  for j = 1:100  <executable>;  end | **Ex 2a.**  for j = 0:5:500  <executable>;  end | **Ex 3a.**  for j = [0 5 3 7 -2]  <executable>;  end |
| **j = 4** | **j = 15** | **j = 7** |

|  |  |
| --- | --- |
| **Ex 1b. Recall the lake example from the Arrays/Matrices Worksheet. Suppose I wanted to see the effect of different rate constants k on the contaminant concentration C over time at a single point in the lake.**  t = 0:3:30; C\_0 = 10;  C = nan(6,length(t)); %preallocation  for k = [0.1 0.3 0.5 0.7 0.9 1.1]  C = C\_0.\*exp(-k.\*t); %C represents the concentration at a point in the lake  end | |
| **What values will be stored in C when this loop is finished running?** | |

How would I save the data from each of the iterations and prevent the previous iterations from being overwritten?

Commonly, your indexing variable is used to, well, index calculated values into your array, ie, it determines the position of your data in the array.

|  |  |  |
| --- | --- | --- |
| **Ex 1c. Building an array**  x = ones(1,20);  for j = 2:2:20  x(j) = 0;  end | **Ex 2c. Contaminant concentration at one point in a lake over time**  C = nan(1,31);  C\_0 = 10; k = 0.1;  for t = 1:31  C(t) = C\_0\*exp(-k\*t);  end | **Ex 3c.**  y = zeros(1,50); %preallocation  y(1) = 15; %set first element  y(2) = 10; %set second element  for j = 3:length(y)  y(j) = (y(j-1)+y(j-2))/2;  end |
| **x(1:5) = [1 0 1 0 1]** | **C(5) = 6.0653** | **y(5) = 11.8750** |

A **while loop** is used when you don’t initially know how many iterations you want, but you want to **iterate through the same commands repeatedly until** **certain conditions are met**. Often, you will have some variables that are set to some initial value. These variables will then change with each iteration.

|  |  |
| --- | --- |
| while <boolean statement>  <executable>  end | To implement a counter for a while loop, set a count variable x = 1 before you start your while loop. Then set x = x+1 inside the while loop. |

Commonly, the value of some variable(s) is changed on each iteration until the boolean statement is no longer true.

|  |  |
| --- | --- |
| **Underline the variable(s) that is/are getting updated each iteration in each of the examples.** | |
| **Ex 1. How many iterations will this script go through? What will the final values of x1 and x2 be?**  x1 = 4; x2 = 0;  while x2-x1 < 0  x2 = x2+1;  x1 = x1-1;  end | **Ex 2. Recall the ideal gas law PV = nRT. A balloon has a maximum volume of 10 m3. This script calculates the maximum moles of gas the balloon can hold at T = 298 K without popping. What are V and n on the third iteration of this script?**  P = 101; V = 1; n = 0.1;  R = 8.314; T = 298;  while V <= 10  V = n\*R\*T/P;  n = n+0.01;  end |
| **x1 = 2**  **x2 = 2**  **3 iterations** | **V = 2.9436**  **n = 0.1300**  **This script goes through 32 iterations** |

**How many iterations does the following script go through?**

**This is an infinite loop ie, it will go through infinite iterations**

|  |  |
| --- | --- |
| **Ex 3.**  x = 1; y = 1;  while z < 1  z = 1/(x+y);  y = 1.1\*y;  end | **To stop any loop:**  **Mac users: command + . or ctrl + c**  **Everyone else: ctrl + c** |

|  |
| --- |
| **Pr 1. How would I save the data from each of the iterations and prevent the previous iterations from being overwritten?**  t = 0:3:30; C\_0 = 10;  **k = [0.1 0.3 0.5 0.7 0.9 1.1];**  C = nan(1,length(t)); %preallocation  for j = **1:length(k)**  **C(j)** = C\_0.\*exp(-**k(j)**.\*t);  end |

1. NOTE: If you don’t suppress statements in your loops, every iteration will print in the command window as MATLAB runs through your loop. [↑](#footnote-ref-0)