MATHCAD'S PROGRAMMING LANGUAGE

The for Loop

Introduction

In this section we look at the *for loop* construct in Mathcad programming. A loop is a means of performing a sequence of similar operations repeatedly over some range of values.

Looping outside of programs can be accomplished with a range variable. When a Mathcad document is treated as a program, the entire worksheet can be iterated by **external scripting of variables**.

Range variables can be used in many ways, such as filling an array,

$i := 0 .. 20$

$X_i := \sin(i)$

accumulating values contained in a specific array,

$sum\_of\_cubes_0 := 0$

$sum\_of\_cubes_{0..i} := sum\_of\_cubes_{0..i} + \left(X_i\right)^3$

$sum\_of\_cubes = (0.769)$

or performing more complicated iterations over large ranges,

$A_0 := 1 \qquad B_0 := 0.5$

$j := 1 .. 1000$

$$\begin{pmatrix} A_j \\ B_j \end{pmatrix} := \begin{pmatrix} \dfrac{A_{j-1}}{2^j} + j \cdot B_{j-1} \\ \exp\left(\mathrm{mod}\left(A_{j-1}, 10\right)\right) - B_{j-1} \end{pmatrix}$$

Range variables can also be used in a programmatic sense with the **until function**, demonstrated earlier.

These computations may be part of a sequence of steps that lend themselves to a program definition. As we mentioned in the section on assignments, defining range variables is not allowed within a program. The construct there is the **for** loop.

The for Loop Operator

Clicking on the **for** button in the Programming toolbar creates an operator that looks like

for ▮ ∈ ▮

▮

Note: As with the **if operator** we emphasize that you cannot type this operator. You must use the button.

Before we give the details, here are some simple uses of the **for** loop in a program.

This function creates an array specifying *N* equal subintervals of the interval [*a*, *b*]

$$\text{Partition}(a,b,N) := \left| \begin{array}{l} \Delta \leftarrow \dfrac{b-a}{N} \\ \text{for } i \in 0..N \\ \quad P_i \leftarrow a + i \cdot \Delta \\ P \end{array} \right.$$

$$\text{Partition}(0,1,5) = \begin{pmatrix} 0 \\ 0.2 \\ 0.4 \\ 0.6 \\ 0.8 \\ 1 \end{pmatrix} \qquad \text{Partition}(0,4,4) = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

The next function adds the reciprocals of the nonzero elements in a vector:

$$\text{rsum}(V) := \left| \begin{array}{l} \text{sum} \leftarrow 0 \\ \text{for } x \in V \\ \quad \text{sum} \leftarrow \text{sum} + \left(\begin{array}{l} 0 \quad \text{if } x = 0 \\ \frac{1}{x} \quad \text{otherwise} \end{array} \right) \\ \text{sum} \end{array} \right.$$

$$\text{rsum}\left(\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}\right) = 1.833 \qquad \text{rsum}\left(\begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}\right) = 1$$

So, there you see **for** in action. Here are the details.

The first placeholder in the for operator,

iteration variable

for ▮ ∈ ▮

▮

is reserved for the name of a variable that we'll refer to as the **iteration variable**:

for i ∈ ▮

▮

for k ∈ ▮

▮

for x ∈ ▮

▮

The second placeholder contains a set of values that the iteration variable will take on over the course of the for loop.

range, list of values, vector

for ▮ ∈ ▮

▮

This set of values can be specified with a range

for $i \in 0 .. 10$

▮

The range may be increasing, decreasing, have a specified increment, or use the default increment (decrement) of 1 (-1).

Alternatively, the set of values may be specified simply as a list of numbers separated by commas.

for $k \in 17, 3, -5, 9$

▮

Lastly, the set of iteration variable values may be the elements of a vector

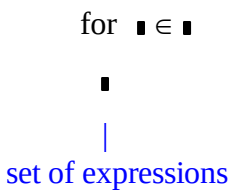
$$V := \begin{pmatrix} 1.2 \\ 9.0 \\ -2.3 \\ 8.1 \end{pmatrix}$$

for $x \in V$

▮

Or some combination of the above may be used, separated by commas.

The final placeholder contains an expression or set of expressions, that is, a subprogram that you wish executed for each value of the loop variable in the specified range.



These will usually be local assignment statements but do not have to be.

Here are some examples of completed **for** loops.

for $i \in 0 .. 10$

$t_i \leftarrow \sqrt{i}$

for $k \in 17, 3, -5, 9$

$sum \leftarrow sum + k$

$prod \leftarrow prod \cdot k$

for $x \in V$

$sign \leftarrow sign + 1 \quad \text{if } x > 0$

$sign \leftarrow sign - 1 \quad \text{if } x < 0$

$sign \leftarrow sign \quad \text{otherwise}$

Execution of a **for** loop then performs the indicated computations in the set of expressions for each value of the loop variable in the order specified. However, this execution is only of interest (or of use, for that matter) within a program.

for $i \in 0 .. 10$ =

$t_i \leftarrow \sqrt{i}$

	0
0	0
1	1
2	1.414
3	1.732
4	2
5	2.236
6	2.449
7	2.646
8	2.828
9	3
10	3.162

You shouldn't need to use the **for** loop outside of a program, mainly because the assignments inside a **for** loop must be local. Range variables in conjunction with worksheet-level vector/matrix assignments should do what is required at the worksheet level. Note that the t assigned above is local only, and not available in the worksheet.

t =

Examples of programs using the **for** loops above are

$$r := \left| \begin{array}{l} \text{for } i \in 0..10 \\ \quad t_i \leftarrow \sqrt{i} \\ t \end{array} \right. \qquad r_0 = 0 \qquad r_1 = 1 \qquad r_5 = 2.236$$

$$Q := \left| \begin{array}{l} \text{sum} \leftarrow 0 \\ \text{prod} \leftarrow 1 \\ \text{for } k \in 17, 3, -5, 9 \\ \quad \left| \begin{array}{l} \text{sum} \leftarrow \text{sum} + k \\ \text{prod} \leftarrow \text{prod} \cdot k \end{array} \right. \\ \begin{pmatrix} \text{sum} \\ \text{prod} \end{pmatrix} \end{array} \right. \qquad Q = \begin{pmatrix} 24 \\ -2.295 \times 10^3 \end{pmatrix}$$

$$p := \left| \begin{array}{l} \text{sign} \leftarrow 0 \\ \text{for } x \in V \\ \quad \left| \begin{array}{l} \text{sign} \leftarrow \text{sign} + 1 \quad \text{if } x > 0 \\ \text{sign} \leftarrow \text{sign} - 1 \quad \text{if } x < 0 \\ \text{sign} \leftarrow \text{sign} \quad \text{otherwise} \end{array} \right. \end{array} \right.$$

$$p = 2$$

In the case of a range specified with a vector, the order of values the iteration variable takes on is from the first element to the last in sequence.

The symbol

$\in$

is usually read as "element of" or simply "in" and **for** loops such as

$\text{for } i \in 0..10$ $\text{for } x \in 1, 1.2..3$

might be read as "for *i* in the set 0 to 10 do ...." and "for *x* from 1 to 3 with steps of 0.2 do the following..." respectively.

Note: The three methods described that define the range of iteration may be combined into a single list. Just separate the pieces with commas.

For example, the **for** loop in

$$\begin{array}{l} q \leftarrow 0 \\ \text{for } w \in \begin{pmatrix} 2.3 \\ 1.5 \\ 0.1 \end{pmatrix}, 5, 8, -1, \begin{pmatrix} 16 \\ 10^2 \end{pmatrix}, (10, 12..20) \\ \quad q \leftarrow q + w \\ q \end{array} = 221.9$$

performs the exact same sequence of iterations as

$$\begin{array}{l} q \leftarrow 0 \\ \text{for } w \in 2.3, 1.5, 0.1, 5, 8, -1, 16, 10^2, 10, 12, 14, 16, 18, 20 \\ \quad q \leftarrow q + w \\ q \end{array} = 221.9$$

or

$$W := \begin{pmatrix} 2.3 & 1.5 & 0.1 & 5 & 8 & -1 & 16 & 10^2 & 10 & 12 & 14 & 16 & 18 & 20 \end{pmatrix}^T$$

$$\begin{array}{l} q \leftarrow 0 \qquad = 221.9 \\ \text{for } w \in W \\ \quad q \leftarrow q + w \\ q \end{array}$$

Note in specifying the **for** values

$$\begin{pmatrix} 2.3 \\ 1.5 \\ 0.1 \end{pmatrix}, 5, 8, -1, \begin{pmatrix} 16 \\ 10^2 \end{pmatrix}, (10, 12 .. 20)$$

we enclosed the range piece

$$10, 12 .. 20$$

in parentheses. Using parentheses is a practice we *highly* recommend. The comma that appears in a range definition means something different from the comma used to separate values in our iteration value list. These different meanings can lead to confusion.

For example at a glance, the list

$$1, 4, 6 .. 10$$

could be a list of the values 1 and 4 followed by the range $6 .. 10$ meaning a total set of values:

$$1 \quad 4 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10$$

or the list could mean the single value 1 followed by the range $4, 6 .. 10$ meaning the set:

$$1 \quad 4 \quad 6 \quad 8 \quad 10$$

Parentheses remove any ambiguity.

$$1, 4, (6 .. 10) \qquad 1, (4, 6 .. 10)$$

Examples Using for Loops

As you might guess, **for** loops are especially useful when working with arrays.

Our first function takes a function of two variables *f* as an argument and an integer *N* and returns an (*N*+1) x (*N*+1) grid of values of *f* over the square

$$0 \le x \le 1 \quad 0 \le y \le 1$$

$$\text{fgrid}(f, N) := \left| \begin{array}{l} \text{for } i \in 0..N \\ \quad \text{for } j \in 0..N \\ \quad\quad S_{i,j} \leftarrow f\left(\frac{i}{N}, \frac{j}{N}\right) \\ S \end{array} \right.$$

$$F(x, y) := \sin(x) + \cos(y)$$

$$\text{fgrid}(F, 5) = \begin{pmatrix} 1 & 0.98 & 0.921 & 0.825 & 0.697 & 0.54 \\ 1.199 & 1.179 & 1.12 & 1.024 & 0.895 & 0.739 \\ 1.389 & 1.369 & 1.31 & 1.215 & 1.086 & 0.93 \\ 1.565 & 1.545 & 1.486 & 1.39 & 1.261 & 1.105 \\ 1.717 & 1.697 & 1.638 & 1.543 & 1.414 & 1.258 \\ 1.841 & 1.822 & 1.763 & 1.667 & 1.538 & 1.382 \end{pmatrix}$$

Note in this program a **for** loop lives inside another **for** loop. There is nothing wrong with that. For the first value of the outer iteration variable

$$i \leftarrow 0$$

the innermost **for** loop is executed in its entirety, that is

$$\text{for } j \in 0..N$$

$$S_{0,j} \leftarrow f\left(\frac{0}{N}, \frac{j}{N}\right)$$

then the next value of the outer iteration variable is used,

$$i \leftarrow 1$$

$$\text{for } j \in 0..N$$

$$S_{1,j} \leftarrow f\left(\frac{1}{N}, \frac{j}{N}\right)$$

and so on. Note that this construct means the array S is filled up a row at a time.

Our next example locates all occurrences of a number in a vector returning a vector of indices or the number -1 if no occurrences are found.

$$\begin{aligned}
&\text{locate}(x, V) := \left|\begin{array}{l}
\text{count} \leftarrow 0 \\
\text{for } i \in 0..\text{last}(V) \\
\quad \text{if } x = V_i \\
\quad\quad \left|\begin{array}{l} I_{\text{count}} \leftarrow i \\ \text{count} \leftarrow \text{count} + 1 \end{array}\right. \\
I \quad \text{if count} > 0 \\
-1 \quad \text{otherwise}
\end{array}\right.
\end{aligned}$$

$$\text{locate}\left[1, \begin{pmatrix} 9 \\ -2 \\ 0 \\ 6 \end{pmatrix}\right] = -1 \qquad \text{locate}\left[0.3, \begin{pmatrix} 0.3 \\ 0.4 \\ 0.3 \\ 0.3 \\ 1 \end{pmatrix}\right] = \begin{pmatrix} 0 \\ 2 \\ 3 \end{pmatrix}$$

Our next example computes the Cartesian product of two vectors. Given two vectors of any sizes, the Cartesian product is the set of all points (vectors of two elements) that can be formed by taking an element from the first vector and an element from the second.

$$\text{CartProd}(X, Y) := \left| \begin{array}{l} \text{ind} \leftarrow 0 \\ \text{for } x \in X \\ \quad \text{for } y \in Y \\ \quad\quad \left| \begin{array}{l} P_{\text{ind}} \leftarrow \begin{pmatrix} x \\ y \end{pmatrix} \\ \text{ind} \leftarrow \text{ind} + 1 \end{array} \right. \\ P \end{array} \right.$$

$$C := \text{CartProd}\left[\begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.1 \\ 0.5 \\ 0.7 \end{pmatrix} \right]$$

$$C^{T} = \left[\begin{pmatrix} 3 \\ 0.1 \end{pmatrix} \begin{pmatrix} 3 \\ 0.5 \end{pmatrix} \begin{pmatrix} 3 \\ 0.7 \end{pmatrix} \begin{pmatrix} 1 \\ 0.1 \end{pmatrix} \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} \begin{pmatrix} 1 \\ 0.7 \end{pmatrix} \right]$$

The two previous examples illustrate how easy it is to define functions that operate on vectors and arrays of unknown size by using for loops.

The following example demonstrates that you can iterate over a vector of matrices. The function **Prod** takes a vector of arrays *V* and returns the array that is the product of all the arrays in *V*.

$$\text{Prod}(V) := \left| \begin{array}{l} P \leftarrow \text{identity}\left(\text{rows}\left(V_0\right)\right) \\ \text{for } A \in V \\ \quad P \leftarrow P \cdot A \\ P \end{array} \right.$$

$$V_0 := \begin{pmatrix} 1 & -4 & 9 \\ -2 & 2 & 1 \\ 6 & 8 & 5 \\ 0 & 0.1 & -5 \\ -1 & 3 & 3 \end{pmatrix} \qquad V_1 := \begin{pmatrix} -0.1 & 1.8 \\ 2.3 & -3.8 \\ 5.1 & 4.4 \end{pmatrix}$$

$$V_2 := \begin{pmatrix} -2.3 \\ 1.7 \end{pmatrix} \qquad V_3 := (-1 \quad 3 \quad 10 \quad 4)$$

$$V_0 \cdot V_1 \cdot V_2 \cdot V_3 = \begin{pmatrix} -12.04 & 36.12 & 120.4 & 48.16 \\ 34.33 & -102.99 & -343.3 & -137.32 \\ 95.51 & -286.53 & -955.1 & -382.04 \\ -20.075 & 60.225 & 200.75 & 80.3 \\ 51.29 & -153.87 & -512.9 & -205.16 \end{pmatrix}$$

$$\text{Prod}(V) = \begin{pmatrix} -12.04 & 36.12 & 120.4 & 48.16 \\ 34.33 & -102.99 & -343.3 & -137.32 \\ 95.51 & -286.53 & -955.1 & -382.04 \\ -20.075 & 60.225 & 200.75 & 80.3 \\ 51.29 & -153.87 & -512.9 & -205.16 \end{pmatrix}$$

More examples of **for** loops appear in the section **Examples of Programming in Mathcad**.