Vectorized Code in 1-D

MATLAB can operate (e.g., perform arithmetic, relational, or logical operations) on entire vectors in one step (in one statement). Code that operate on vectors, instead of on scalars, in one statement is said to be vectorized.

\[ x= [10 \ 20 \ 30] \]
\[ y= [2 \ 1 \ 2] \]
\[ z= [2; 3; 2] \] % column

% Vectorized addition, subtraction
\[ x+y \] % [12 21 32]
\[ x-y \] % [8 19 28]
\[ x+5 \] % [15 25 35]

% Vectorized multiplication, division, power
% Need DOT OPERATOR (.)
\[ x.*y \] % [20 20 60]
\[ x./y \] % [5 10 15]
\[ x.^y \] % [100 20 900]
\[ x.*5 \] % [50 100 150]

% Shape is important!
\[ x+z \] % ERROR! x is a row while z is column
\[ x+z' \] % [12 23 32]

Note: MATLAB is designed to support a field of mathematics called linear algebra. After you learn linear algebra (not in CS100!), you can really harness the power of MATLAB’s matrix computation capabilities. In CS100, we will not use “matrix multiplication” as defined in linear algebra and coded in MATLAB as \( m*n \) where \( m \) and \( n \) are vectors (or matrices). Rather, we multiply the vectors “cell-by-cell” using the MATLAB code \( m.*n \), which means to perform the operation \( m(i)\times n(i) \) for all index \( i \) for vectors \( m \) and \( n \) (assuming that they have the same shape and length). The result of “dot multiply” is a vector the same shape and length as \( m \) and \( n \), as illustrated in the example above.

Example: Pair-sums

Given a vector \( x \), calculate the pair sums and store them in a vector. For example, the vector \([2 \ 1 \ 1 \ 8]\) has pair sums \([3 \ 2 \ 9]\).

Example: How many ’M’s

Write a code fragment to determine how many times the character ’M’ appears in a string \( s \).
Pre-allocating vectors vs. building vectors by concatenation

% Add vectors a and b of same length
n= length(a);
c= zeros(1,n);  % unnecessary statement but improves performance
for k= 1:n
    c(k)= a(k) + b(k);
end

Logical arrays and operations

Logical arrays, i.e., arrays containing logical values, are the results of relational or logical operations. In MATLAB, logical values are zero for false and one (or any non-zero value) for true. Logical values are not just numbers—they have the logical property attached to the data, see the workspace window under “class” when you have a logical value in the MATLAB workspace.

elev = 8*rand(4,3) + 10  % example, elevations on a map
elev > 16  % returns a logical array

% 1-d examples
vec = elev(1,:);  % 1st row of matrix elev
L = vec>16  % logical array indicating result from vec>16
vecHigh = vec(L)  % extract just the cells with values > 16
vecHigh = vec(vec>16)  % combine last two statements in one
                       % this shortcut works for VECTORS only, not matrices
ind = find(vec>16)  % get the indices where vec>16
vecHigh = vec(ind)  % extract just the cells with values > 16

% Create a vector same as vec above except that all the values below 16 are "zeroed out".
L = (vec>16)  % a LOGICAL vector
vecHigh = zeros(1,length(vec))
vecHigh(L) = vec(L)  % assign only to the cells with logical value 1

ind = find(vec>16)  % a vector of INDICES
vecHigh = zeros(1,length(vec))
vecHigh(ind) = vec(ind)  % assign only to the cell numbers stored in ind

% There is a simpler solution using vectorized multiplication...

% 2-d examples
L = elev>16  % logical array (matrix)
elevHigh = elev(elev>16)  % a VECTOR!!!
[ri,ci] = find(elev>16)  % ri is vector that stores row index where elev>16
                       % ci is vector that stores col index where elev>16