# MATLAB Operations

## Common MATLAB operations

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classical Notation</th>
<th>MATLAB notation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assignment (=) Operation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalar</td>
<td>( x = 1.6 )</td>
<td>( x = 1.6 )</td>
</tr>
<tr>
<td>Row vector</td>
<td>( a = [2 \ 3] )</td>
<td>( a = [2 \ 3] )</td>
</tr>
<tr>
<td>Matrix</td>
<td>( \mathbf{A} = \begin{bmatrix} 4 &amp; 5 \ -3 &amp; 2 \end{bmatrix} )</td>
<td>( \mathbf{A} = [4 \ 5; \ -3 \ 2] )</td>
</tr>
</tbody>
</table>

### Arithmetic Operations

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classical Notation</th>
<th>MATLAB notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum - scalars and arrays</td>
<td>( z = x + y )</td>
<td>( z = x+y )</td>
</tr>
<tr>
<td>Difference - scalars and arrays</td>
<td>( z = x - y )</td>
<td>( z = x-y )</td>
</tr>
<tr>
<td>Multiplication - scalars and arrays of consistent dimension</td>
<td>( z = xy )</td>
<td>( z = x*y )</td>
</tr>
<tr>
<td>Multiplication - arrays elementwise</td>
<td>( z(i) = x(i) y(i), \text{ each } i )</td>
<td>( z = x.*y )</td>
</tr>
<tr>
<td>Division – scalars</td>
<td>( z = x / y )</td>
<td>( z = x/y )</td>
</tr>
<tr>
<td>Division - arrays elementwise</td>
<td>( z(i) = x(i) / y(i), \text{ each } i )</td>
<td>( z = x./y )</td>
</tr>
<tr>
<td>Exponentiation –scalars</td>
<td>( z = x^y )</td>
<td>( z = x^y )</td>
</tr>
<tr>
<td>Exponentiation -arrays elementwise</td>
<td>( z(i) = x(i)^{y(i)}, \text{ each } i )</td>
<td>( z = x.^y )</td>
</tr>
</tbody>
</table>

### Sequences stored as arrays

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classical Notation</th>
<th>MATLAB notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increment by 1</td>
<td>5, 6, 7, 8 (array)</td>
<td>5:8</td>
</tr>
<tr>
<td>Increment by 0.3</td>
<td>3, 3.3, 3.6, 3.9 (array)</td>
<td>3:0.3:4</td>
</tr>
<tr>
<td>Selected elements of an array</td>
<td>a(3), a(5), a(7), a(9)</td>
<td>a(3:2:9)</td>
</tr>
</tbody>
</table>

### Simple Functions of arrays

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classical Notation</th>
<th>MATLAB notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine, Cosine (elementwise)</td>
<td>( \sin(x), \cos(x) )</td>
<td>( \sin(x), \cos(x) )</td>
</tr>
<tr>
<td>Exponential (elementwise)</td>
<td>( e^x )</td>
<td>( \exp(x) )</td>
</tr>
<tr>
<td>Maximum</td>
<td>( \max(x) )</td>
<td>( \max(x) )</td>
</tr>
<tr>
<td>Square root (elementwise)</td>
<td>( \sqrt{x} )</td>
<td>( \sqrt{x} )</td>
</tr>
</tbody>
</table>

### Recursive computations and loops
Statements inside a for loop are repeated a specified number of times.

Example 1:

```matlab
for i = 1:5
    m(i) = i*2;
end
```

Returns:

```
m = 2 4 6 8 1
```

Example 2:

```matlab
for i=1:3:11
    m=i;
end
```

Returns:

```
m = 10
```

Remember to always end the loop with end.

Example 3: 1D translation of a particle in a time-dependent velocity field:

```matlab
delta_t=.1;          % define time step
x(1:41)= zeros;      % initialize x
x(1) = 2;
for t=1:40           % begin time loop
    v(t)=10*cos(t);
    x(t+1) = x(t) + v(t)*delta_t;  % update x(t)
end  % end time loop
plot(0:40,x)
```

Translating equations to program commands

Exercise: Equations describing oxygen consumed by decaying organic matter in a stream:

Solution for oxygen conc. (Streeter-Phelps Eq):

\[
c = c_0 - K_1 L_0 \begin{bmatrix}
\frac{-K_d x}{U} & \frac{-K_a x}{U} & \frac{-K_a x}{U}
\end{bmatrix}
\]

\[
K_1 = \frac{K_d}{K_a - K_d}
\]

Plot oxygen depletion vs. distance in a stream (oxygen.m)

Some relevant MATLAB functions: axis
Parameter Values:

- Kd = 0.8, organic matter decay rate, day^(-1)
- Ka = 1.0, aeration rate, day^(-1)
- cs = 8, saturation concentration dissolved oxygen, mg/L
- c0 = 6, upstream concentration dissolved oxygen, mg/L
- L0 = 10, concentration of oxygen-consuming organic matter injected at x=0, mg/L
- U = 4, stream velocity km/day

1. State the problem clearly (as if you were writing it up for someone else to solve). What do you want to calculate? What procedures will you rely upon?

2. Itemize the input and output information needed for the calculation and determine how you want the answers to be displayed (e.g. sketch plots similar to those you wish to produce)

3. Write down a detailed solution procedure. You may want to work out a simplified version of the problem by hand or with a calculator (for some typical inputs) to clarify the operations that need to be carried out.

4. Translate the solution into MATLAB code, correct syntax errors identified by MATLAB, make sure program reproduces your manual solution, debug as required. Keep things as simple as possible at first. If you have problems, simplify even more.

5. Solve the problem two ways: first using vector notation only, and second using a for loop.

6. Test the MATLAB program with different input data and extend the program as desired once it seems to be working.