1.017/1.010 Recitation 2
MATLAB Operations

Common MATLAB operations

Feature	Classical Notation	MATLAB notation
Assignment (=) Operation		
Scalar	$x = 1.6$	`x = 1.6`
Row vector	$a = [2\ \ 3]$	`a = [2  3]`
Matrix	$A = \begin{bmatrix} 4 & 5 \\ -3 & 2 \end{bmatrix}$	`A = [4 5; -3 2]`
Arithmetic Operations		
Sum - scalars and arrays	$z = x + y$	`z = x+y`
Difference - scalars and arrays	$z = x - y$	`z = x-y`
Multiplication - scalars and arrays of consistent dimension	$z = xy$	`z = x*y`
Multiplication - arrays elementwise	$z(i) = x(i)\, y(i)$, each i	`z = x.*y`
Division – scalars	$z = x / y$	`z = x/y`
Division - arrays elementwise	$z(i) = x(i) / y(i)$, each i	`z = x./y`
Exponentiation –scalars	$z = x^{y}$	`z = x^y`
Exponentiation -arrays elementwise	$z(i) = x(i)^{y(i)}$, each i	`z = x.^y`

Feature	Classical Notation	MATLAB notation
Sequences stored as arrays		
Increment by 1	5, 6, 7, 8 (array)	`5:8`
Increment by 0.3	3, 3.3, 3.6, 3.9 (array)	`3:0.3:4`
Selected elements of an array	a(3), a(5), a(7), a(9)	`a(3:2:9)`
Simple Functions of arrays		
Sine, Cosine (elementwise)	$\sin(x)$, $\cos(x)$	`sin(x), cos(x)`
Exponential (elementwise)	e^{x}	`exp(x)`
Maximum	$\max(x)$	`max(x)`
Square root (elementwise)	$\sqrt{x}$	`sqrt(x)`

Recursive computations and loops

Statements inside a `for` loop are repeated a specified number of times.

Example 1:

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

Returns:

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

Example 2:

```
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:

```
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):

$$c = c_s - K_1 L_0 \left[e^{-\frac{K_d x}{U}} - e^{-\frac{K_a x}{U}} \right] - (c_s - c_0) e^{-\frac{K_a x}{U}}$$

$$K_1 = \frac{K_d}{K_a - K_d}$$

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 injetced 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