MATLAB

MATrix LABoratory
hw08 due today
exam4 4/18–4/20
hw09 due 4/21 (check your email)
Interface
Start MATLAB either at the command line, `matlab`, or by clicking the icon.

More access instructions on course site, FAQ.
Why MATLAB?

- Designed for engineering.
- Excellent documentation and toolboxes.
- Ideal applications:
  - Linear algebra
  - Control dynamics
  - Numerical analysis
  - Image processing
What is MATLAB?

- Programming language + environment.
- Proprietary, owned and maintained by MathWorks.
- Dates from late 1970s, under active development.
- Was an influence on NumPy/MPL, so will be familiar.
Basics

- Literals, variables, operators
  
  4 ^ 3

- Expressions
  
  a = 3 * 2
  b = 1 + a

- Semicolon suppresses output (mutes):
  
  ;
  b = b + 2;

- ans is default result.
  
  a / 4

- disp displays the value only.
  
  disp( ans );
MATLAB implements:
- integers
- floating-point numbers
- complex numbers

in 8-, 16-, 32-, and 64-bit versions.

whos shows type, value of all variables in workspace.
Arrays are the fundamental type in MATLAB:

```matlab
a = [ 1 2 3 ];
b = a( 1 );
```
Arrays are the fundamental type in MATLAB:

\[
a = [ 1 \ 2 \ 3 ];
\]

- Arrays are indexed using parentheses:

\[
b = a( 1 );
\]

- MATLAB indexes from one, not zero!
More dimensional arrays use semicolons to separate rows:

\[ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}; \]

Arrays are indexed using parentheses and commas:

\[ a = A(1,2); \]

Helper functions are available:

\[ B = \text{ones}(3,3) + \text{eye}(3,3) + \text{zeros}(3,3); \]
Which of the following will produce this array?

A [ 1 1 1 ] ; [ 2 2 2 ]
B [ 1 1 1 ; 2 2 2 ]
C [ 1 2 ] ; [ 1 2 ] ; [ 1 2 ]
D [ 1 2 ; 1 2 ; 1 2 ]
E [ [ 1 1 1 ] , [ 2 2 2 ] ]
Which of the following will produce this array?

A \[\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}\]

B \[\begin{bmatrix} 1 & 1 & 1 ; 2 & 2 & 2 \end{bmatrix}\star\]

C \[\begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 1 & 2 \end{bmatrix}\]

D \[\begin{bmatrix} 1 & 2 ; 1 & 2 ; 1 & 2 \end{bmatrix}\]

E \[\begin{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} & \begin{bmatrix} 2 & 2 & 2 \end{bmatrix} \end{bmatrix}\]
A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}

Which of the following will access 4 in this array?

A. A(1,0)
B. A[2,1]
C. A(2,1)
D. A(1)(0)
A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}

Which of the following will access 4 in this array?

A. \text{A}(1,0)  
B. \text{A}[2,1]  
C. \text{A}(2,1)  
D. \text{A}(1)(0)
Array operations

% basic mathematics:
A = ( ones( 3,3 ) + 1 ) / 2
B = sin( ones( 3,3 ) * pi )
C = B' % transpose with ' 

% matrix multiplication:
D = eye( 3,4 ) * ones( 4,5 ) * pi
Which of the following will produce this array?

A. \(3 \times \text{ones}(2,2) - 2 \times \text{eye}(2,2)\)
B. \(2 \times \text{ones}(2,2) + \text{eye}(2,2)\)
C. \(3 \times \text{ones}(2,2) - \text{eye}(2,2)\)
D. \(\text{ones}(2,2) + \text{eye}(2,2)\)
Question

\[
\begin{pmatrix}
2 & 1 \\
1 & 2
\end{pmatrix}
\]

Which of the following will produce this array?

A 3*ones( 2,2 ) - 2*eye( 2,2 )
B 2*ones( 2,2 ) + eye( 2,2 )
C 3*ones( 2,2 ) - eye( 2,2 )
D ones( 2,2 ) + eye( 2,2 )
% concatenating arrays
A = [ eye( 3,4 ), eye( 3,5 );
     ones( 2,4 ), ones( 2, 5) ]
How can we produce this array?

A \[[ [1 \ 3 \ 5] \ [2 \ 4 \ 6] ]\]
B \[[ [1 \ 2] \ [3 \ 4] \ [5 \ 6] ]\]
C \[[ [1 \ 3 \ 5] ; [2 \ 4 \ 6] ]\]
D \[[ [1 \ 2] ; [3 \ 4] ; [5 \ 6] ]\]
How can we produce this array?

A \[ \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \]

B \[ \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \]

C \[ \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix} \]

D \[ \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \] *
MATLAB uses .m files for two purposes: scripts and functions.

Comments are indicated as follows:

```matlab
% this is a comment
%
%{
%   this is a block comment
%
```
Use the built-in editor to create these.
Make sure you have the correct working directory.
Scripts contain regular commands in order of execution.
Functions

- Functions must be located in a file of the same name as the function.

```matlab
function [ output ] = function_name( input )
    % ...
end
```

- No explicit `return` statements—rely on values in output variable list.
\[ T_F = \frac{180}{100} T_C + 32 \]

File TempC2F.m:

```matlab
function [ Tf ] = TempC2F( Tc )
    Tf = Tc * ( 180/100 ) + 32;
end
```
Strings

- Indicated with single quotes (only!).
  
  ```
  s = 'XFEM';
  ```

- Print formatted strings with `sprintf`:
  ```
  sprintf( '%f %f', sin(pi/3), cos(pi/4) );
  ```
“Matrix dimensions must agree.”

It is necessary to distinguish elementwise operations and matrix operations.

\[
A = 2 * \text{ones}(2,2) \\
B = A .* \text{eye}(2,2) \\
C = A * \text{eye}(2,2)
\]

These are distinguished by a dot \( . \) in front of the operator.
%% loop through time steps
for i = 1:2:10
    disp( sprintf( 'The number is %i.' , i ) )
end
The for loop ranges over a set of possible values.
The for loop ranges over a set of possible values.
This is not as flexible as Python’s in syntax—think of always having to loop over the index rather than the item.
Ranges are straightforward: 1:10, 1:2:10, 0.1:0.1:0.5. Also have linspace available.
We create a for loop as follows:

- `statement for var in range`, where you create `var` and provide `range`
- one or more statements
- closing statement `end`
We create a for loop as follows:

- statement for var in range, where you create var and provide range
- one or more statements
- closing statement end

Also have continue and break available.
function [ y ] = absolute( x )
    y = 0;
    if x >= 0
        y = x;
    else
        y = -x;
    end
We create an if/else statement as follows:
- the keyword if
- a logical comparison *(more on these!)*
- a block of code
We create an if/else statement as follows:
- the keyword if
- a logical comparison *(more on these!)*
- a block of code
- the keyword elseif *(note this!)*
- a new logical comparison
- a different block of code
We create an `if/else` statement as follows:
- the keyword `if`
- a logical comparison *(more on these!)*
- a block of code
- the keyword `elseif` *(note this!)*
- a new logical comparison
- a different block of code
- the keyword `else`
- a different block of code
We create an if/else statement as follows:
- the keyword if
- a logical comparison *(more on these!)*
- a block of code
- the keyword elseif *(note this!)*
- a new logical comparison
- a different block of code
- the keyword else
- a different block of code
- the keyword end
MATLAB does not have a bool data type.
MATLAB does not have a bool data type.

Instead of True/False, MATLAB uses integers:

- 0 means False
- 1 means True
MATLAB does **not** have a `bool` data type.

Instead of True/False, MATLAB uses integers:
- 0 means False
- 1 means True

Available logical operators include:

- `<`, `>`, `<=`, `>=`, `==`, `!=`
- `&&` for ‘and’, `||` for ‘or’
- `ismember` checks equality of elements in arrays.
- Also, logical operators work as indices!
MATLAB does not have a bool data type.
Instead of True/False, MATLAB uses integers:
- 0 means False
- 1 means True
Available logical operators include:
- <, >, <=, >=, ==, 
- && for ‘and’, || for ‘or’
- ismember checks equality of elements in arrays.
Also, logical operators work as indices!
A( A<0 )
Like NumPy, but no imports (anywhere).
Remember: parentheses, indexing from 1.
Hard to do dict-like things, easy to do numpy-like operations.
Next steps
Next steps

- hw08 due today
- exam4 4/18–4/20
- hw09 due 4/21 (check your email)