MATLAB

Introduction

CS101 Lecture #22

2016-11-16

Administrivia

Administrivia

- Homework #10 is due Tuesday, Dec. 20.
- Homework #11 is due Weds, Jan. 4.
- Midterm #2 is Monday, Dec. 19 from 7–10 p.m.

Warmup Quiz

Question #1

```
import numpy as np
tmax = 10.0
dt = 0.01
nt = int( tmax/dt ) + 1
x = np.zeros( (nt,) )
for i in range( 0,dt ):
    vx = x[ i-1 ] / np.sin( i )
    x[ i+1 ] = x[ i ] + vx * dt
```

Which uncaught error will halt this code?

A `ZeroDivisionError`

B `TypeError`

C `SyntaxError`

D `IndexError`

Question #1

```
import numpy as np
tmax = 10.0
dt = 0.01
nt = int( tmax/dt ) + 1
x = np.zeros( (nt,) )
for i in range( 0,dt ):
    vx = x[ i-1 ] / np.sin( i )
    x[ i+1 ] = x[ i ] + vx * dt
```

Which uncaught error will halt this code?

A `ZeroDivisionError`

B `TypeError` ⋆(range error)

C `SyntaxError`

D `IndexError`

Question #2

```
x = np.ones( 10 )
for i in range( 10 ):
    try:
        ???
    except:
        print( 'Error on step %d.'%err )
        continue
```

Which of the following candidates for `???` would *not* produce an error message?

A `x += x[ i+1 ]`
B `x[ i ] /= 0`
C `x[ -i-1 ] = sum( x[ :i ] )`
D `x[ 10-i ] = sum( x[ :i ] )`

Question #2

```
x = np.ones( 10 )
for i in range( 10 ):
    try:
        ???
    except:
        print( 'Error on step %d.'%err )
        continue
```

Which of the following candidates for `???` would *not* produce any error message?

A `x += x[ i+1 ]` index error

B `x[ i ] /= 0` ⋆(surprise!)

C `x[ -i-1 ] = sum( x[ :i ] )` ⋆

D `x[ 10-i ] = sum( x[ :i ] )` index error

Error Handling

Common exceptions

- `SyntaxError`
- `NameError`
- `TypeError`
- `ValueError`
- `IOError`
- `IndexError`
- `KeyError`
- `ZeroDivisionError`
- `IndentationError`
- `Exception`

MATLAB

Why MATLAB?

- Designed for engineering.
- Excellent documentation: MATLAB Central.
- Ideal applications:
  - Linear algebra
  - Control dynamics
  - Numerical analysis
  - Image processing
- Many toolboxes available.

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 );
```

Numeric types

- 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.

Array types

- Arrays are the fundamental type in MATLAB:

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

- Arrays are indexed using parentheses:

```
b = a( 1 );
```

- MATLAB counts from one, not zero!

Multidimensional arrays

- More dimensional arrays use semicolons to separate rows:

```
A = [ 1 2 3 ; 4 5 6 ];
```

- Arrays are indexed using parentheses and commas:

```
a = A( 1,2 );
```

- Helper functions are available:

```
B = ones( 3,3 ) + eye( 3,3 ) + zeros( 3,3 );
```

Question

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{pmatrix}$$

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 ] ]
```

Question

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{pmatrix}$$

Which of the following will produce this array?

A `[ 1 1 1 ] ; [ 2 2 2 ]`

B `[ 1 1 1 ; 2 2 2 ]` $\star$

C `[ 1 2 ] ; [ 1 2 ] ; [ 1 2 ]`

D `[ 1 2 ; 1 2 ; 1 2 ]`

E `[ [ 1 1 1 ] , [ 2 2 2 ] ]`

Question

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

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 )`

Question

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Which of the following will access 4 in this array?

A `A( 1,0 )`
B `A[ 2,1 ]`
C `A( 2,1 )` $\star$
D `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
```

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 )`

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 )` $\star$

Array operations

```
% concatenating arrays
A = [  eye( 3,4 ),  eye( 3,5 );
      ones( 2,4 ), ones( 2, 5) ]
```

Question

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

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 ] ]`

Question

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

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 ] ]` ⋆

Scripting

- MATLAB uses `.m` files for two purposes: scripts and functions.
- Comments are indicated as follows:

```
% this is a comment
%{
  this is a block comment
%}
```

Scripting

- 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.

```
function [ output variables ] = function_name(
    % ...
end
```

- No explicit `return` statements—rely on values in output variable list.

Functions

$$T_{\mathsf{F}} = \frac{180}{100} T_{\mathsf{C}} + 32$$

File `TempC2F.m`:

```
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 v. element operations

- “Matrix dimensions must agree.”
- It is sometimes necessary to distinguish *elementwise* operations and *matrix* operations.

```
A = 2 * ones( 2,2 )
B = A .* eye( 2,2 )
C = A  * eye( 2,2 )
```

- These are distinguished by a dot `.` in front of the operator.
- We won’t emphasize this but frequently you must distinguish.