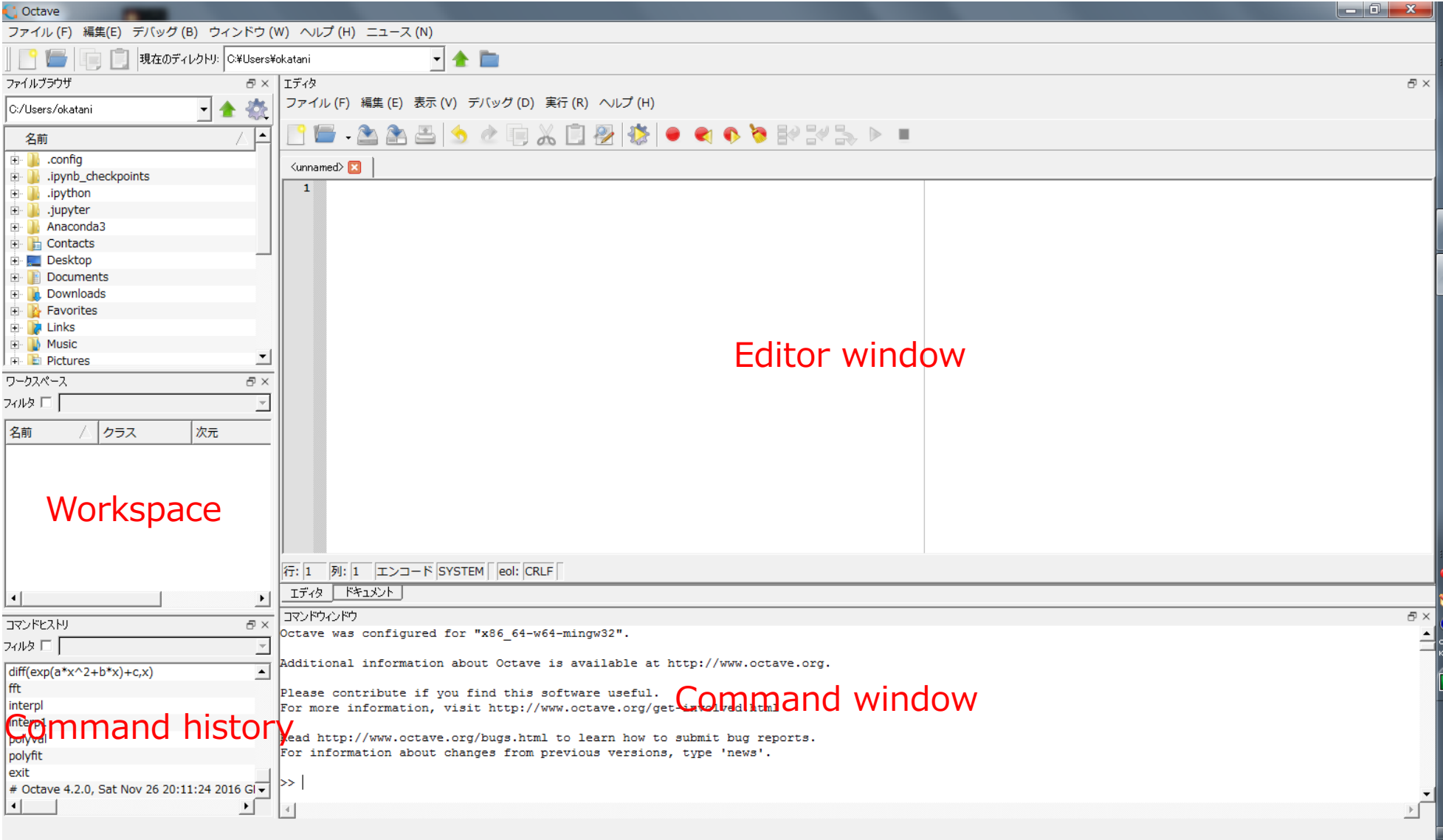


2. Fundamentals of Octave (&MATLAB)

- Octave GUI(Graphical User Interface)
- Command Window
- Scripts
- Variables
- Matrices
- Arithmetic operations & special values
- Mathematical functions
- Input/output with files
- Loops
- Conditional branch & flow control
- Plotting graphs

Octave GUI



Using Command Window

- Example: Type "1+2" and press the Enter key after the prompt ">>"

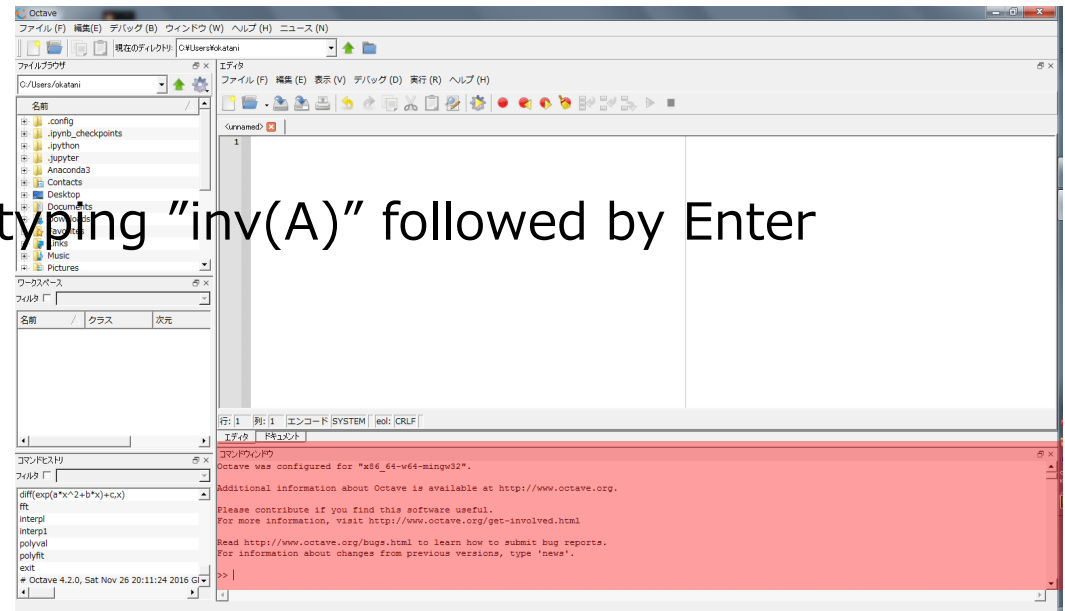
```
>> 1+2  
ans = 3  
>>
```

- You can create a 2x2 matrix A by typing as follows:

```
>> A=[1,2;3,4]  
A =  
    1    2  
    3    4
```

- You can calculate its inverse by typing "inv(A)" followed by Enter

```
>> inv(A)  
ans =  
 -2.00000  1.00000  
  1.50000 -0.50000
```



Writing a script file

- Type as follows in the Editor window, select “Save File”-“File” in the Editor window menu, type “hello”, and click “Save”
 - The script should be saved as “hello.m”
- Type “hello” followed by Enter to run the contents
 - Same as choosing “Save File and Run”-“Run” in the menu

The screenshot shows the Octave software interface. The main editor window displays a script file named 'hello.m' with the following code:

```
1 A=[1,2;3,4]
2 inv(A)
```

The command window below the editor shows the output of the script:

```
A =
  1  2
  3  4
ans =
 -2.00000  1.00000
  1.50000 -0.50000
```

Red boxes and arrows highlight the code in the editor and the corresponding output in the command window.

Using variables

- You can create and use a variable like `A` in the earlier example
 - The name of a variable should be different from existing files and variables
 - There is no limitation in the length of variable names; it must be less than 19 characters in MATLAB, though

```
>> the_1st_variable=[1;2];  
>> the_1st_variable  
the_1st_variable =  
  
    1  
    2
```

- Numeric characters and `'_'` (underscore) can be used for variable names
- Result won't be displayed by typing `';'`(semicolon) at the end

- All the variables you created so far will be displayed in Workspace
- You can remove a variable with the data by typing `clear`

```
>> clear A
```

Using matrices

- The most fundamental data representation in Octave/Matlab
- A matrix of any size can be created by using ',' to separate elements and ';' to separates rows;

2x3 matrix

```
>> A=[1,2,3;2,3,4]
A =
     1     2     3
     2     3     4
```

3x2 matrix

```
>> B=[1,2;2,3;3,4]
B =
     1     2
     2     3
     3     4
```

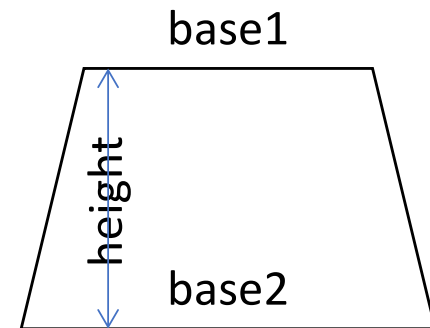
- You can get the size of a matrix using a built-in function `size`

```
>> size(A)
ans =
     2     3
>> size(B)
ans =
     3     2
```

Arithmetic operation and special values

- Basic operators : +, -, *, /

```
>> base1=3.0;base2=5.0;height=3.0;  
>> area=(base1+base2)*height/2  
area = 12
```



- Exponentiation : ^

```
>> 2^40  
ans = 1.0995e+12
```

- π

```
>> pi  
ans = 3.1416
```

- Imaginary unit : i or j

```
>> i  
ans = 0 + 1i  
>> j  
ans = 0 + 1i  
>> exp(-pi*i)  
ans = -1.0000e+00 - 1.2246e-16i
```

$$e^{i\pi} = -1$$

(Euler's formula)

Mathematical functions

- Trigonometric functions
 - sin, sinh, asin, cos, cosh, acos, tan, tanh, atan, atan2
- Exponential, log functions, etc.
 - exp, log, log10, sqrt
- Various operations on matrix elements
 - sum, max, min, sort, mod
- Absolute value and complex numbers
 - abs, conj, imag, real

```
>> sin(pi/2)
ans = 1
>> sin(pi)
ans = 1.2246e-16
>> log(e)
ans = 1
```

```
>> A
A =
     1     2     3
     2     3     4
>> sum(A)
ans =
     3     5     7
>> sum(sum(A))
ans = 15
```

```
>> a=2.0-3.0j
a = 2 - 3i
>> imag(a)
ans = -3
>> real(a)
ans = 2
>> abs(-a)
ans = 3.6056
>> conj(a)
ans = 2 + 3i
```


Input and output with files

- You can write the value of a variable into a specified file:

```
>> save('A.txt', 'A')
```

- Then read the written value from the file:

```
>> load('A.txt')
>> A
A =
     1     2     3
     2     3     4
```

```
>> B=load('A.txt')
>> B
B =
     1     2     3
     2     3     4
```

- You can also save/load the whole contents of Workspace into/from a specified file

```
>> save('workspace1')
```

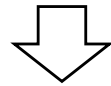
```
>> load('workspace1')
```

Loops

- Repeat a series of commands with `for index=start:step:end ... end`

Script

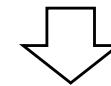
```
# loop1.m
for i=1:10
    x = 2^i;
    printf('%d: %f\n', i, x)
end
```



Result

```
>> loop1
1: 2.000000
2: 4.000000
3: 8.000000
4: 16.000000
5: 32.000000
6: 64.000000
7: 128.000000
8: 256.000000
9: 512.000000
10: 1024.000000
```

```
# loop2.m
# calculate position of a vehicle
# with a constant acceleration
a = 1.0; # acceleration
for t=0.0:0.5:3 # time
    y=.5*a*t^2; # position
    printf('%f: %f\n', t, y)
end
```



```
>> loop2
0.000000: 0.000000
0.500000: 0.250000
1.000000: 1.000000
1.500000: 2.250000
2.000000: 4.000000
2.500000: 6.250000
3.000000: 9.000000
```

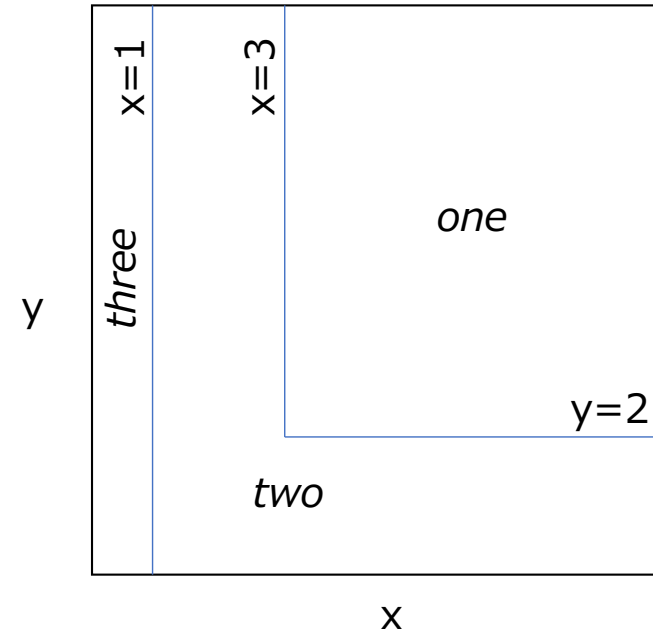
Conditional branch & flow control

- if-elseif-else-end structure

Script

```
#ifelse.m
if x > 3.0 && y > 2.0
    disp('one')
elseif x > 1.0
    disp('two')
else
    disp('three')
end
```

Logical AND
(if both are true)



```
#ifelse2.m
if x < 3.0 || y < 2.0
    if x < 1.0
        disp('three')
    else
        disp('two')
    end
else
    disp('one')
end
```

Logical OR
(if either is true)



Results

```
>> x=4;y=5;
>> ifelse
one
>> x=2;y=5;
>> ifelse
two
>> x=y=0;
>> ifelse
three
```

Plotting a graph

- `plot(x,y)`, where `x` is a vector of length `m` storing `x` coordinates and `y` is a vector of the same length storing `y` coordinates

```
>> x=-pi:pi/100:pi;  
>> y=x.^2;  
>> plot(x,y)
```

`).^` expresses
squaring each element

- To plot different curves in a single graph

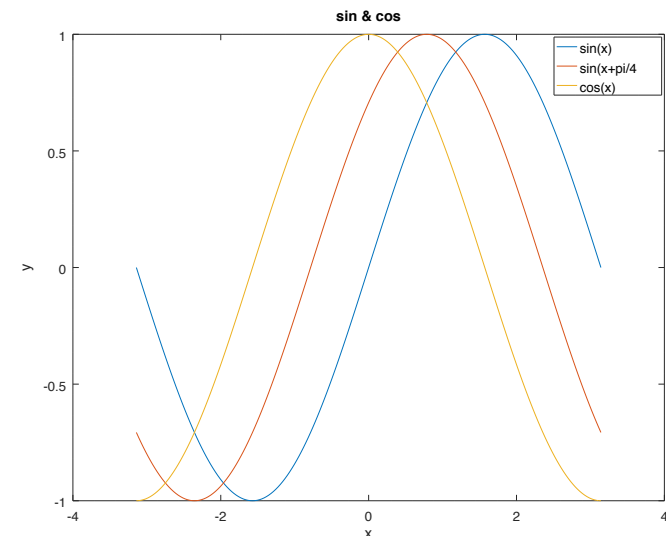
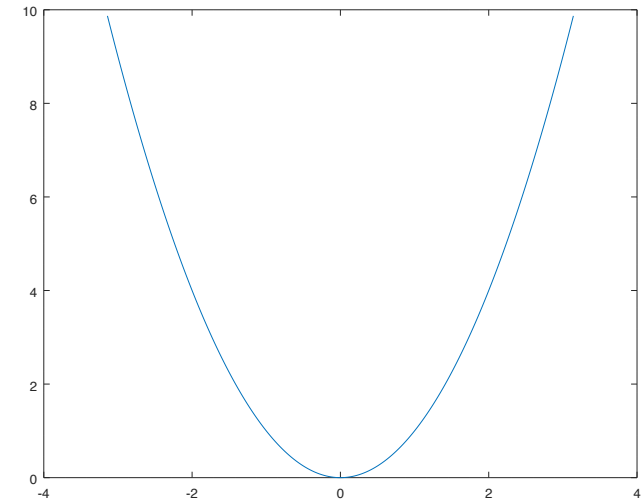
```
>> plot(x,sin(x),x,sin(x+.25*pi),x,cos(x))
```

- To set axis labels, titles, and legends

```
>> xlabel('x'), ylabel('y'), title('sin & cos')  
>> legend('sin(x)', 'sin(x+pi/4)', 'cos(x)')
```

- To change font sizes (before calling `plot`)

```
>> set(0,"defaultaxesfontsize",20)  
>> set(0,"defaulttextfontsize",20)
```




Exercises 2.1 (assignments)

- Find all numbers of 3 digits such that the sum of the cubes of its digits equals the number itself; an example is 153, because $1^3+5^3+3^3 = 153$
- Revise the script below to find these numbers

```
for i = 100:999
    i1 = mod(i, 10);
    i2 = mod(floor(i/10), 10);
    i3 = floor(i/100);
    disp([i3 i2 i1])
end
```

Hint: This script scans every three-digit number and gets its three digits

- Write a script that finds the same numbers in a different way by filling in the blanks below:

```
for i3 = 1:9
    for i2 = 0:9
        for i1 = 0:9
            
        end
    end
end
```