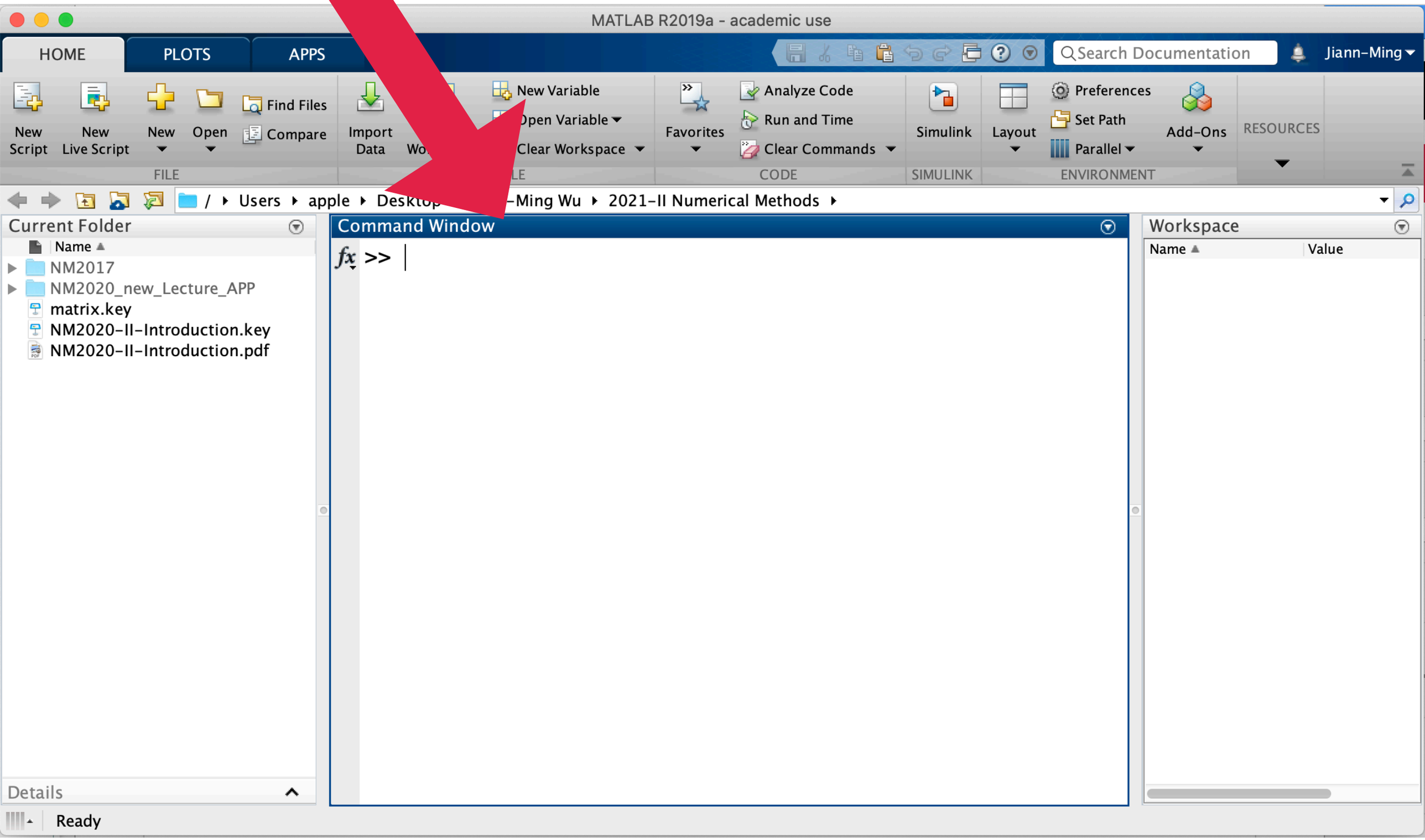
Matlab

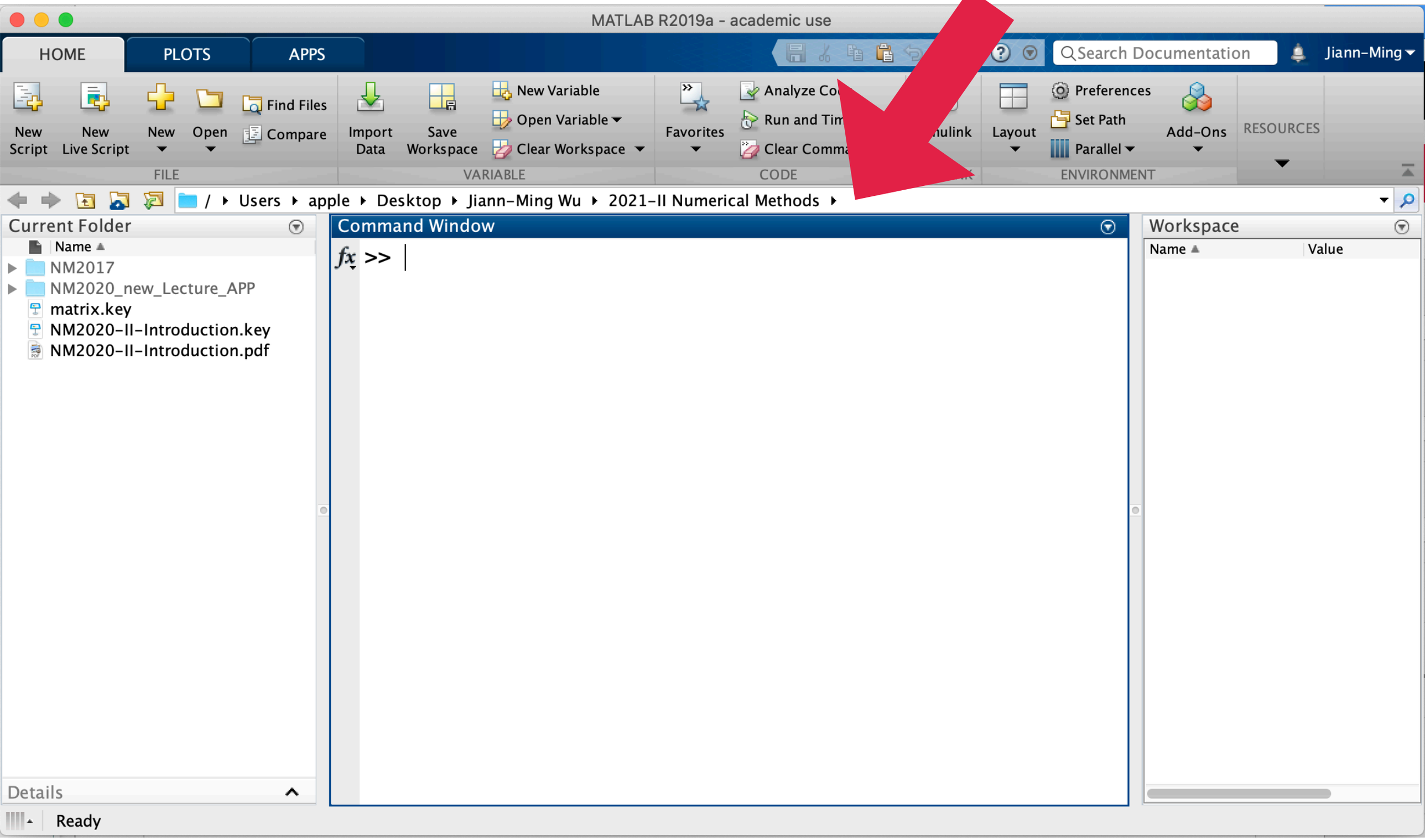
Matrix

Jiann-Ming Wu 2021

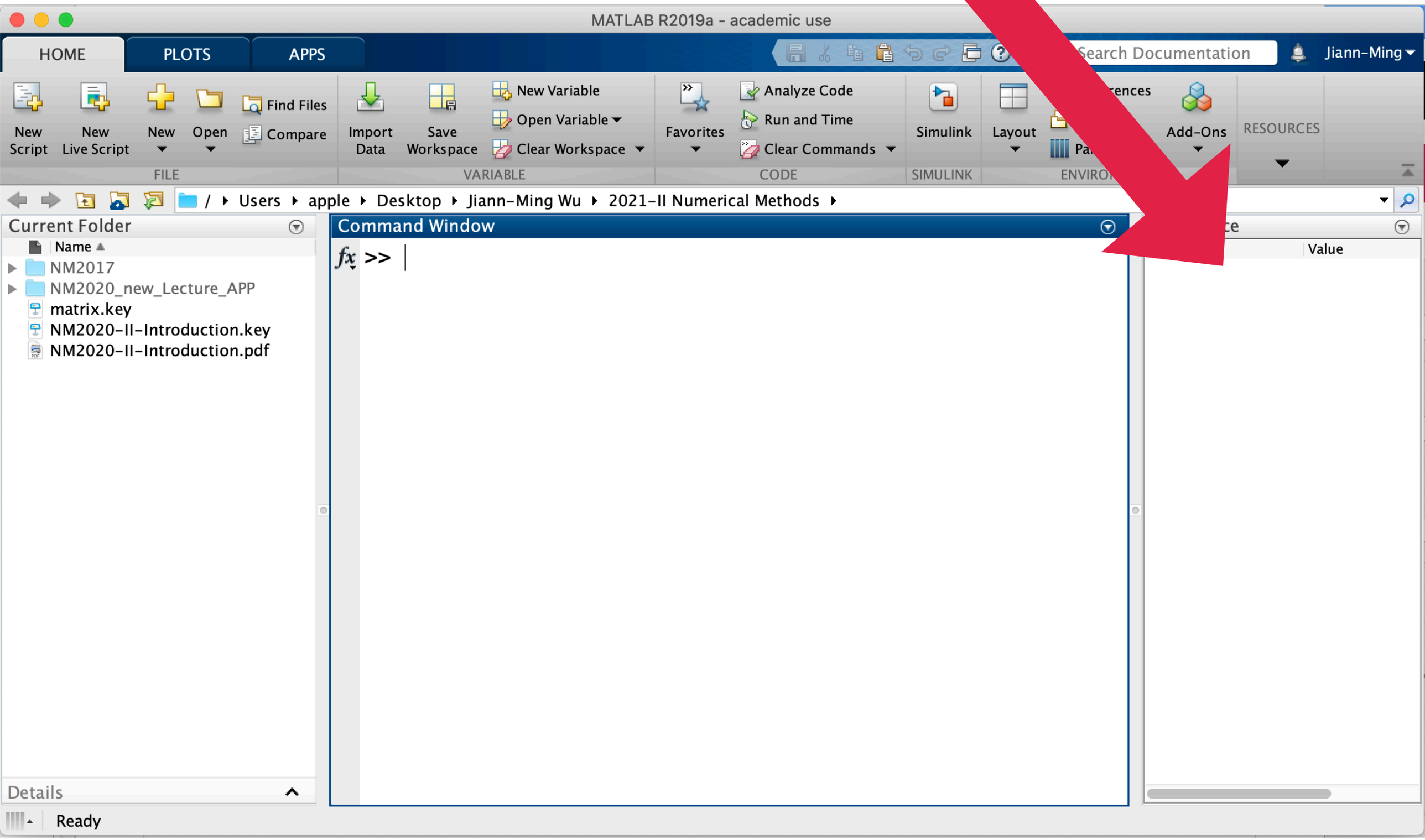
Command window



Current directory



Workspace


MATLAB R2019a - academic use
HOME
PLOTS
APPS
Search Documentation
Jiann-Ming
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Favorites
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Add-Ons
RESOURCES
FILE
VARIABLE
CODE
SIMULINK
/ ▸ Users ▸ apple ▸ Desktop ▸ Jiann-Ming Wu ▸ 2021-II Numerical Methods ▸
Current Folder
Name
NM2017
NM2020_new_Lecture_APP
matrix.key
NM2020-II-Introduction.key
NM2020-II-Introduction.pdf
Details
Command Window
>>
Value
Ready

Defining Matrices in Matlab

Command Window

```
>> A = [2 4 6;8 10 12]

A =

     2     4     6
     8    10    12

>> B = [11 9; 7 5;3 1]

B =

    11     9
     7     5
     3     1
```

';' implies new row

```
A =

     2     4     6
     8    10    12
```

```
B =

    11     9
     7     5
     3     1
```

```
>> size(A)

ans =

     2     3
```

Column number of A equals row number of B

```
>> size(B)

ans =

     3     2
```

A * B is defined

Matrix operations

- Standard matrix multiplication
- Element-wise matrix operations
- Extracting sub-matrices
- Transpose
- Concatenation

Matrix multiplication

Element-wise multiplication

Matrix multiplication

```
A =

     2     4     6
     8    10    12
```

```
B =

    11     9
     7     5
     3     1
```

```
>> A*B

ans =

    68    44
   194   134
```

Matrix multiplication

```
A =

     2     4     6
     8    10    12
```

```
B =

    11     9
     7     5
     3     1
```

```
>> B*A

ans =

    94   134   174
    54    78   102
    14    22    30
```

Column number of B equals row number of A

Element-wise multiplication

$$
\begin{array}{rc}
time = & [0.5 \quad 0.7 \quad 0.5 \quad 1.1 \quad 2.0] \\
 & \quad * \quad * \quad * \quad * \quad * \\
speed = & [15 \quad 16 \quad 18 \quad 15 \quad 14] \\
 & \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
distance = & [7.5 \quad 11.2 \quad 9.0 \quad 16.5 \quad 28]
\end{array}
$$

```
>> time = [0.5 0.7 0.5 1.1 2.0]

time =

    0.5000    0.7000    0.5000    1.1000    2.0000

>> speed = [15 16 18 15 14]

speed =

    15    16    18    15    14

>> distance = time * speed
Error using  *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the
first matrix matches the number of rows in the second matrix. To perform elementwise
multiplication, use '.*'.
```

Element-wise multiplication

```
>> time

time =

    0.5000    0.7000    0.5000    1.1000    2.0000

>> speed

speed =

    15    16    18    15    14

>> distance = time .* speed

distance =

    7.5000   11.2000    9.0000   16.5000   28.0000
```

Element-wise multiplication

$$KE = \frac{1}{2}mv^2$$

```
>> m = 155 + 20

m =

    175

>> ke = 0.5*m*speed.^2

ke =

    1.0e+04 *

     1.9688    2.2400    2.8350    1.9688    1.7150
```

Element-wise square

MAtrix Operations
reshape
row
column
sub-matrix
concatenation
transpose

reshape

- A= [1 2 3 4; 5 6 7 8]
- reshape(1:8,4,2)'

MATLAB Mobile

```
>> A= [1 2 3 4; 5 6 7 8]

A =

     1     2     3     4
     5     6     7     8
```

MATLAB Mobile

```
>> reshape(1:8,4,2)'

ans =

     1     2     3     4
     5     6     7     8
```

Rows

- >> A(1,:)

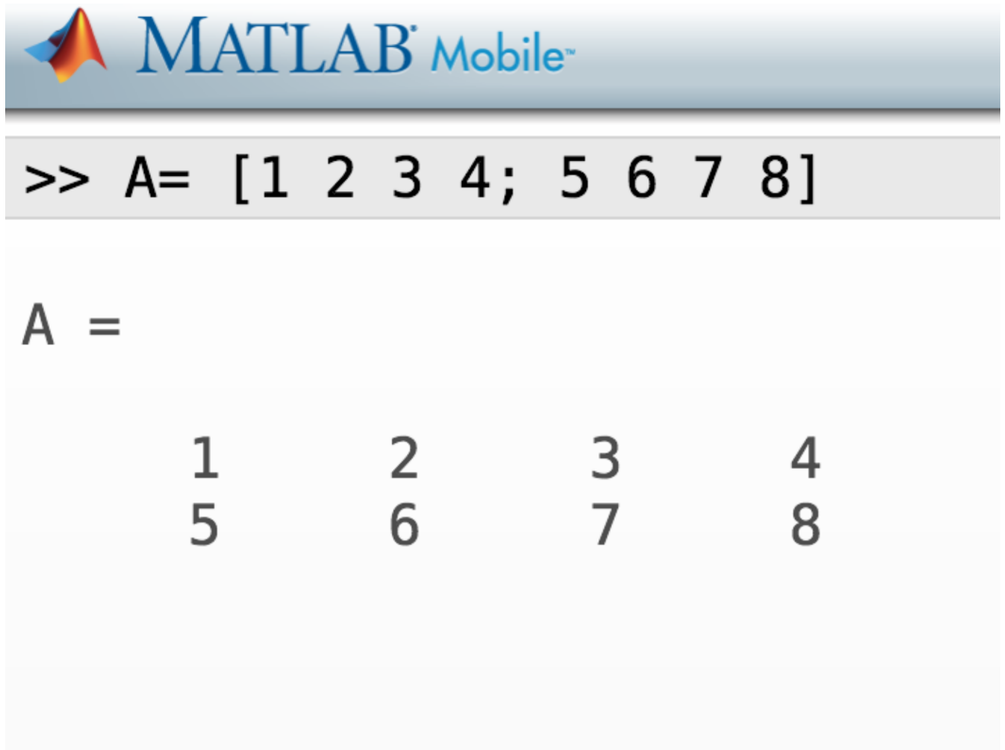
ans =

1 2 3 4

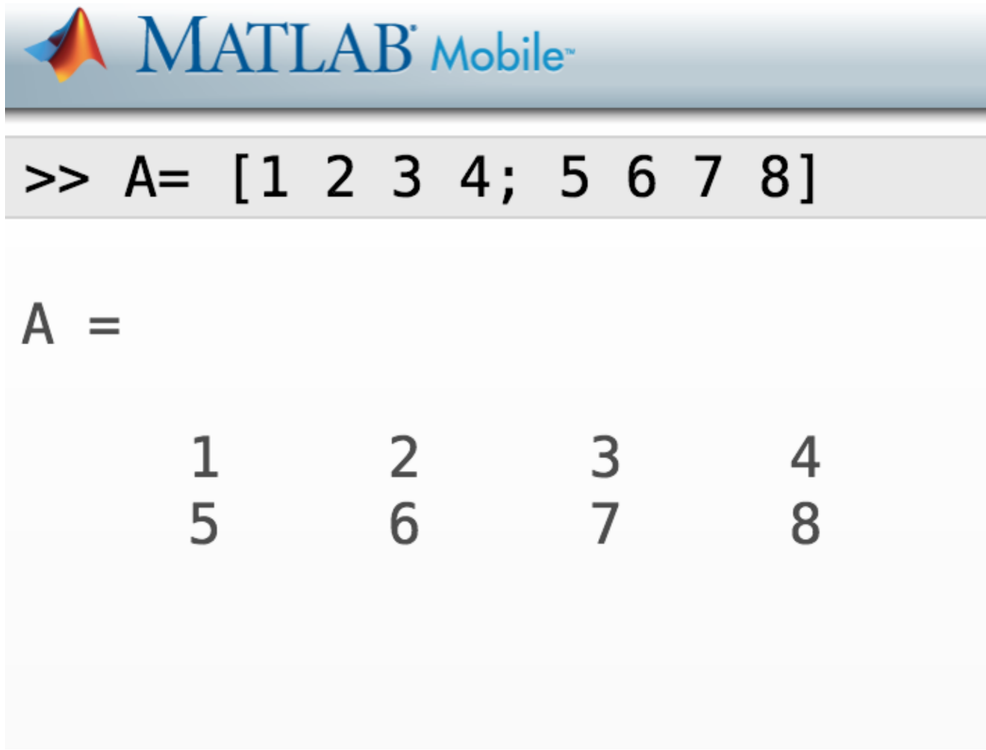
- >> A(2,:)

ans =

5 6 7 8



Columns



- >> A(:,1)

ans =

1

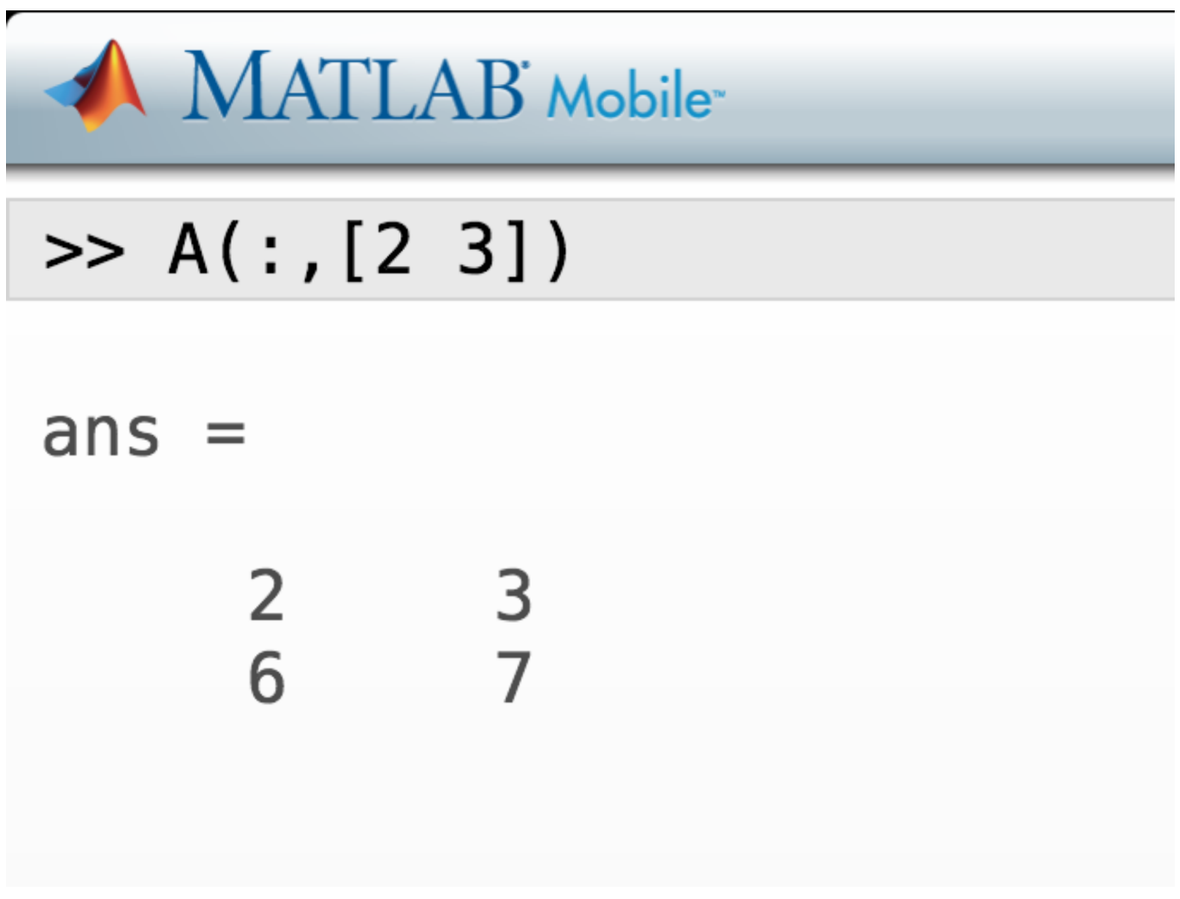
5

- >> A(:,[2 3])

ans =

2 3

6 7

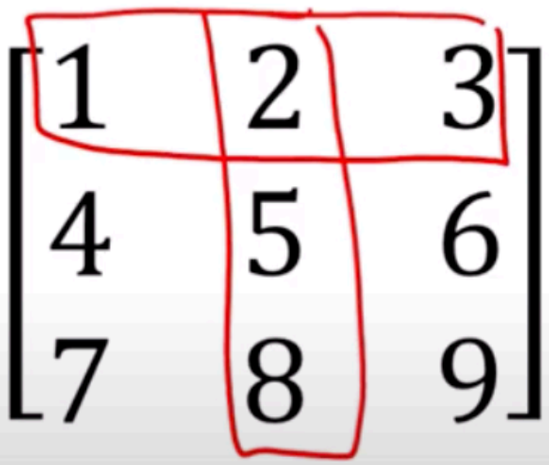


single element

entire row or column

submatrix

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$



View By:
Category | Product List

Using MATLAB
MATLAB

Using Simulink
Simulink
Physical Modeling
Event-Based Modeling
Real-Time Simulation and Testing

Workflows
Parallel Computing
Reporting and Database Access
Systems Engineering
Code Generation
Application Deployment
Verification, Validation, and Test
Cloud Capabilities

Documentation

Other

Using MATLAB

MATLAB

Using Simulink

Simulink

Physical Modeling

Event-Based Modeling

Applications

AI, Data Science, and Statistics

Mathematics and Optimization

Signal Processing

Image Processing and Computer Vision

Control Systems

Resources

Release Notes

Installation Help

Hardware Support

Community

File Exchange

MATLAB

The Language of Technical Computing

Millions of engineers and scientists worldwide use MATLAB® to analyze and design the systems and products transforming our world. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets, and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems.

Release Notes
PDF Documentation

Get Started

Learn the basics of MATLAB

Language Fundamentals

Syntax, array indexing and manipulation, data types, operators

Data Import and Analysis

Import and export data, including large files; preprocess data, visualize and explore

Language Fundamentals

Syntax, array indexing and manipulation, data types, operators

MATLAB is an abbreviation for "matrix laboratory." While other programming languages usually work with numbers one at a time, MATLAB® operates on whole matrices and arrays. Language fundamentals include basic operations, such as creating variables, array indexing, arithme data types.

Categories

Entering Commands

Build and run MATLAB statements

Matrices and Arrays

Array creation, combining, reshaping, rearranging, and indexing

Data Types

Numeric arrays, characters and strings, tables, structures, and cell arrays; data type conversion

Operators and Elementary Operations

Arithmetic, relational, and logical operators, special characters, rounding, set functions

Loops and Conditional Statements

Control flow and branching using keywords, such as `if`, `for`, and `while`

Matrices and Arrays

Array creation, combining, reshaping, rearranging, and indexing

R2023b

Matrices and arrays are the fundamental representation of information and data in MATLAB®. You can create common arrays and grids, combine existing arrays, manipulate an array's shape and content, and use indexing to access array elements. For an overview of matrix and array manipulation, watch Working with Arrays.

Creating, Concatenating, and Expanding Matrices

The most basic MATLAB® data structure is the matrix. A matrix is a two-dimensional, rectangular array of data elements arranged in rows and columns. The elements can be numbers, logical values (`true` or `false`), dates and times, strings, `categorical` values, or some other MATLAB data type.

Even a single number is stored as a matrix. For example, a variable containing the value 100 is stored as a 1-by-1 matrix of type `double`.

```
A = [12 62; 93 -8]
```

```
A = 2×2

    12    62
    93    -8
```

Array Indexing

R2023b

In MATLAB®, there are three primary approaches to accessing array elements based on their location (index) in the array. These approaches are indexing by position, linear indexing, and logical indexing.

Open in MATLAB Online

```
A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
```

```
A = 4×4

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

```
e = A(3,2)
```

```
e = 10
```

Removing Rows or Columns from a Matrix

R2023b

The easiest way to remove a row or column from a matrix is to set that row or column equal to a pair of empty square brackets `[]`. For example, create a 4-by-4 matrix and remove the second row.

Open in MATLAB Online

```
A = magic(4)
```

```
A = 4×4

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
A(2,:) = []
```

```
A = 3×4

    16     2     3    13
     9     7     6    12
     4    14    15     1
```

Reshaping and Rearranging Arrays

R2023b

Many functions in MATLAB® can take the elements of an existing array and put them in a different shape or sequence. This can be helpful for preprocessing your data for subsequent computations or analyzing the data.

Open in MATLAB Online

```
A = [1 4 7 10; 2 5 8 11; 3 6 9 12]
```

```
A = 3×4

     1     4     7    10
     2     5     8    11
     3     6     9    12
```

```
B = reshape(A,2,6)
```

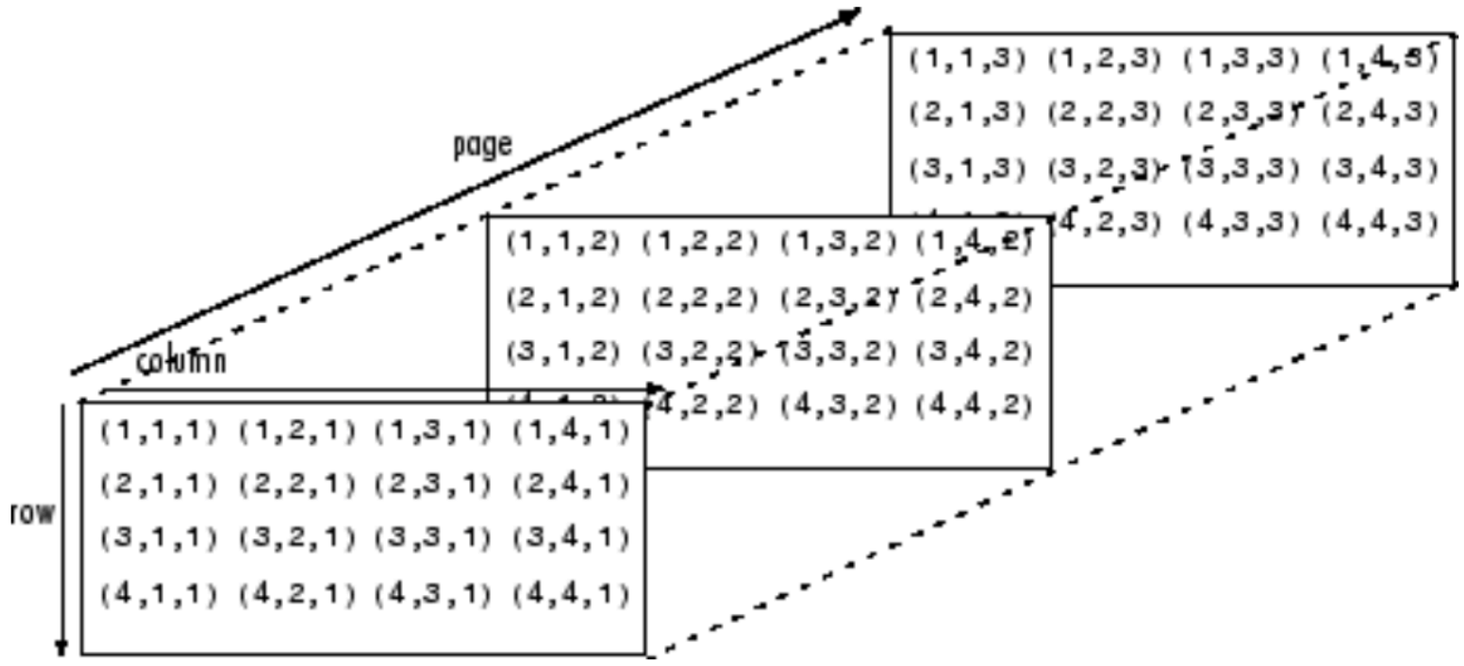
```
B = 2×6

     1     3     5     7     9    11
     2     4     6     8    10    12
```

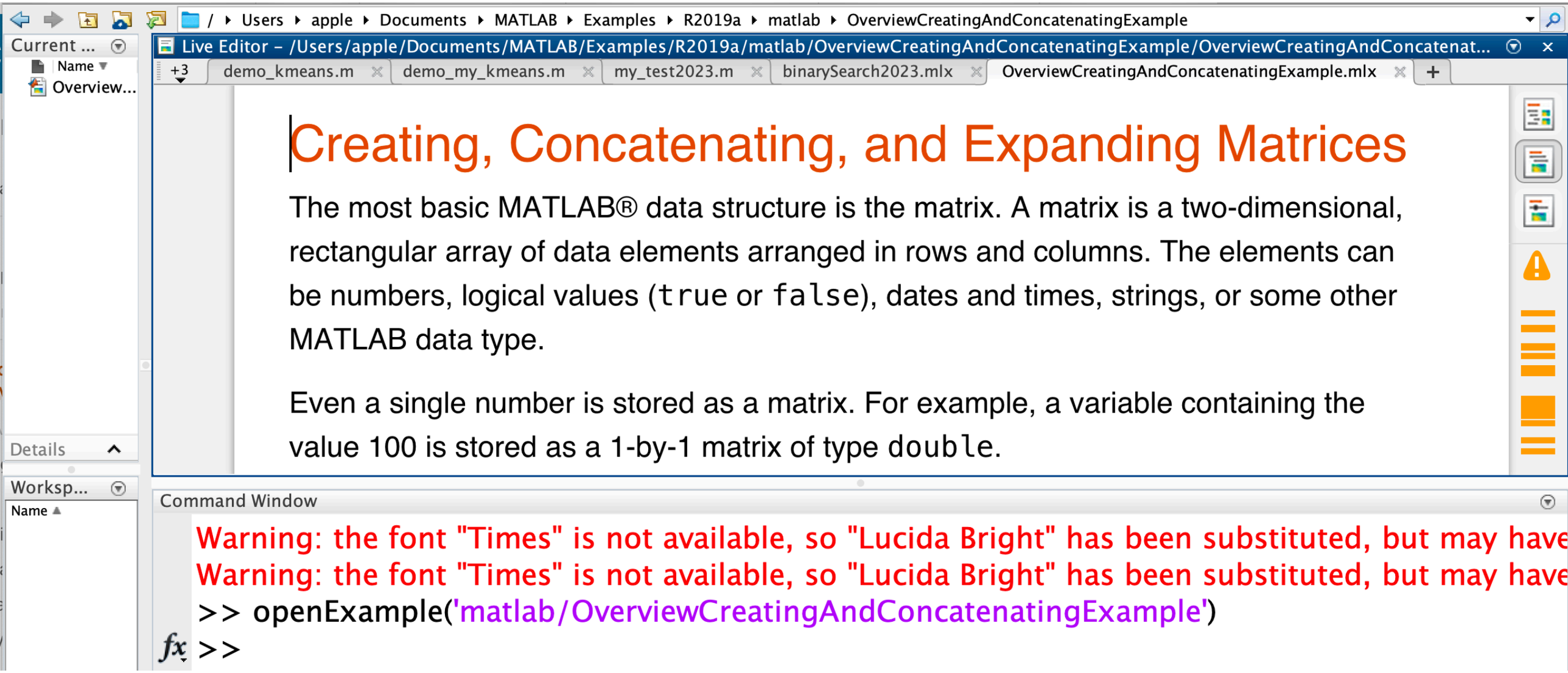
Multidimensional Arrays

R2023b

A multidimensional array in MATLAB® is an array with more than two dimensions. In a matrix, the two dimensions are represented by rows and columns.

Open in MATLAB Online




/ ▸ Users ▸ apple ▸ Documents ▸ MATLAB ▸ Examples ▸ R2019a ▸ matlab ▸ OverviewCreatingAndConcatenatingExample
Current ...
Name
Overview...
Live Editor - /Users/apple/Documents/MATLAB/Examples/R2019a/matlab/OverviewCreatingAndConcatenatingExample/OverviewCreatingAndConcatenat...
+3
demo_kmeans.m
demo_my_kmeans.m
my_test2023.m
binarySearch2023.mlx
OverviewCreatingAndConcatenatingExample.mlx
Creating, Concatenating, and Expanding Matrices
The most basic MATLAB® data structure is the matrix. A matrix is a two-dimensional, rectangular array of data elements arranged in rows and columns. The elements can be numbers, logical values (true or false), dates and times, strings, or some other MATLAB data type.
Even a single number is stored as a matrix. For example, a variable containing the value 100 is stored as a 1-by-1 matrix of type double.
Details
Worksp...
Name
Command Window
Warning: the font "Times" is not available, so "Lucida Bright" has been substituted, but may have
Warning: the font "Times" is not available, so "Lucida Bright" has been substituted, but may have
>> openExample('matlab/OverviewCreatingAndConcatenatingExample')
>>

Row, column and sub-matrix

```
>> C = reshape(1:9,3,3)'

C =

     1     2     3
     4     5     6
     7     8     9

>> C(1,2)

ans =

     2

>> C(1,:)

ans =

     1     2     3

>> C(:,2)

ans =

     2
     5
     8
```

single element

entire row or column

submatrix

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>> C(2:3,1:2)

ans =

     4     5
     7     8
```

row index is a vector, [2 3]

Column index is a vector, [1 2]

Concatenation

- Vertical concatenation

New line for placing row 2

```
>> row1 = [1 2 3 4]

row1 =

     1     2     3     4

>> row2 = [5 6 7 8]

row2 =

     5     6     7     8


>> E = [row1;
        row2]

E =

     1     2     3     4
     5     6     7     8
```

Concatenation

- Horizontal concatenation

Concatenate two columns horizontally

```
>> col1 = [ 1; 2; 3]

col1 =

     1
     2
     3

>> col2 = [ 4; 5; 6]

col2 =

     4
     5
     6

>> F = [ col1 col2]

F =

     1     4
     2     5
     3     6
```

Transpose

row i is the same as column i of A

```
>> A = magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> transpose(A)

ans =

    16     5     9     4
     2    11     7    14
     3    10     6    15
    13     8    12     1

>> A'

ans =

    16     5     9     4
     2    11     7    14
     3    10     6    15
    13     8    12     1
```

Useful built-in matrix functions

ones
zeros
eye
diag
rand
magic
sum
size
length
transpose

zeros, ones

```
>> zeros(4,6)

ans =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

All elements are zero

```
>> ones(4,6)

ans =

     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
```

All elements are one

eye, diag

```
>> eye(5)

ans =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1

>> diag([1 2 3 4 5])

ans =

     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5
```

An identical matrix

A diagonal matrix

rand, magic

Uniform sample from [0, 1]

```
>> rand(3,4)

ans =

    0.8147    0.9134    0.2785    0.9649
    0.9058    0.6324    0.5469    0.1576
    0.1270    0.0975    0.9575    0.9706
```

Sum of elements at each row equals sum of elements at each column

```
>> magic(5)

ans =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

sum

```
>> A = magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> sum(A)

ans =

    34    34    34    34

>> sum(A,2)

ans =

    34
    34
    34
    34
```

Sum of elements of each column

Sum of elements of each row

length, size

```
>> a = 0:9

a =

     0     1     2     3     4     5     6     7     8     9

>> length(a)

ans =

    10
```

```
>> D = ones(4,5)

D =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>> size(D)

ans =

     4     5
```

linspace

```
>> v = linspace(0,1,5)

v =

         0    0.2500    0.5000    0.7500    1.0000
```

values equally spce [0, 1]

Summary

1. Defining matrices in Matlab.

2. Matrix operations.
   a) Standard matrix multiplication
   b) Elementwise matrix operations
   c) Extracting submatrices
   d) Transpose
   e) Concatenation

3. Useful built-in matrix functions.
   a) zeros, ones, eye, diag, rand
   b) Spaced vectors ([ ], linspace)
   c) size, length

Matrix operations

- Define matrix
- Sub-matrices
- Reproduction
- Reshape
- Determinant, inversion

Sub-matrix

- a=1:1:36
- A=reshape(a,4,9)
- % Extract rows 2:4 and columns 5:8 of A
- A(2:4,5:8)

MATLAB Mobile

```
>> A(2:4,5:8)

ans =

    18    22    26    30
    19    23    27    31
    20    24    28    32
```

MATLAB Mobile

```
>> A

A =

     1     5     9    13    17    21    25    29    33
     2     6    10    14    18    22    26    30    34
     3     7    11    15    19    23    27    31    35
     4     8    12    16    20    24    28    32    36
```

Reshape

```
v=1:12;
reshape(v,3,4)

ans =

     1     4     7    10
     2     5     8    11
     3     6     9    12
```

Reshape

- a=1:1:36
- m=4;n=9
- A=reshape(a,4,9)
- % reshape vector a to matrix A
- % A is composed of m rows and n columns

drive.matlab.com

Facebook | Sign in to MATLAB Drive - MATLAB & Simulink - MathWorks

MATLAB® Drive

MATLAB® Drive

MathWorks®

Email

No account? Create one!

By signing in you agree to our privacy policy.

Next

Gray image

cmw2.bmp

```
I=imread('cmw2.bmp');
imshow(I)
```

```
 I=imread('cmw2.bmp');
image(I)
```

MATLAB Mobile

```
>> dir

.           .session    Shared
..          Published   cmw2.bmp
```

```
>> I=imread('cmw2.bmp');
>> image(I)
```

Sub-matrices

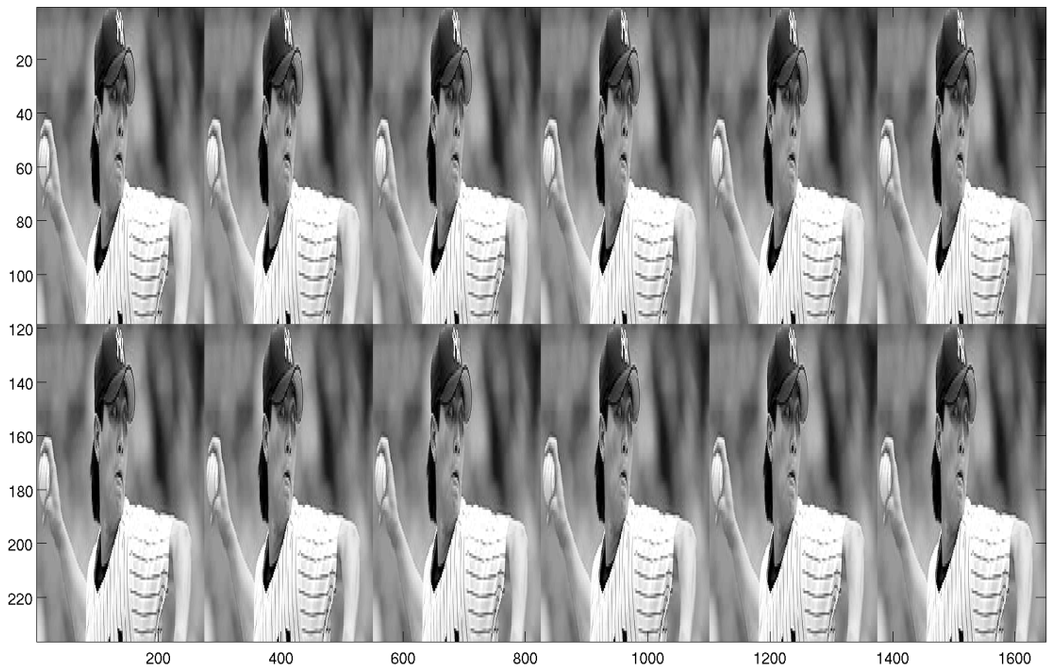
```
I=imread('cmw2.bmp');
image(I);
[m,n]=size(I);
m2=ceil(m/2);
n2=ceil(n/3*2);
Is=I(1:m2,1:n2); figure
imagesc(Is)
colormap(gray)
```

MATLAB Mobile™

```
>> I=imread('cmw2.bmp');
image(I);
[m,n]=size(I);
m2=ceil(m/2);
n2=ceil(n/3*2);
Is=I(1:m2,1:n2); figure
imagesc(Is)
colormap(gray)
```

Repeat

```
I2=repmat(Is,2,3);
imagesc(I2);
colormap(gray)
```



Repmat

- A=[1 2;3 4]
- m=2;n=3;
- repmat(A,m,n)
- % repeat A vertically m times
- % repeat the result horizontally n times

MATLAB Mobile

```
>> A=[1 2;3 4]
m=2;n=3;
repmat(A,m,n)
```

```
A =

     1     2
     3     4


ans =

     1     2     1     2     1     2
     3     4     3     4     3     4
     1     2     1     2     1     2
     3     4     3     4     3     4
```

Get Rows

- A=reshape(1:16,4,4)
- a=3;b=1;c=2;
- A([a b c],:)
- % get rows specified by [a b c]

```
>> A=reshape(1:16,4,4)
a=3;b=1;c=2;
A([a b c],: )

A =

     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16


ans =

     3     7    11    15
     1     5     9    13
     2     6    10    14
```

Get Columns

- A=reshape(1:16,4,4)
- a=3;b=1;c=2;
- A(:,[a b c])
- % get columns specified by [a b c]

```
>> A=reshape(1:16,4,4)
a=3;b=1;c=2;
A(:,[a b c])
```

```
A =

     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16


ans =

     9     1     5
    10     2     6
    11     3     7
    12     4     8
```

Row exchange

```
V=reshape(1:12,3,4)
V([3 2],:)=V([2 3],:)
```

V =

1	4	7	10
3	6	9	12
2	5	8	11

Column exchange

```
V=reshape(1:12,3,4)

V(:,[2 3])=V(:,[3 2])
```

V =

1	7	4	10
2	8	5	11
3	9	6	12

Column sum

```
V=reshape(1:12,3,4)

V =

     1     4     7    10
     2     5     8    11
     3     6     9    12

>> sum(V)

ans =

     6    15    24    33
```

Column sum

```
V=reshape(1:12,3,4)

V =

     1     4     7    10
     2     5     8    11
     3     6     9    12

>> sum(V,1)

ans =

     6    15    24    33
```

Row sum

```
V=reshape(1:12,3,4)

V =

     1     4     7    10
     2     5     8    11
     3     6     9    12
```

```
>> sum(V,2)

ans =

    22
    26
    30
```

Mean of columns

```
V=reshape(1:12,3,4)

V =

     1     4     7    10
     2     5     8    11
     3     6     9    12

>> mean(V)

ans =

     2     5     8    11
```

Mean of rows

```
V=reshape(1:12,3,4)

V =

     1     4     7    10
     2     5     8    11
     3     6     9    12

>> mean(V,2)
```

Inversion

```
A=reshape(1:4,2,2);
B=inv(A)

B =

   -2.0000    1.5000
    1.0000   -0.5000
```

Linear system

$$2x + y - z = 1$$

$$-3x - 2y + 5z = 0$$

$$x + y + z = 5$$

```
A = [2 1 -1;-3 -2 5;1 1 1];
b =[1; 0; 5];
ans = inv(A)*b
mean(abs(A*ans - b)) < 10^-14
```

$$\begin{bmatrix} 2 & 1 & -1 \\ -3 & -2 & 5 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}$$

```
A =

     2     1    -1
    -3    -2     5
     1     1     1

b =

     1
     0
     5
```

```
>> ans = inv(A)*b

ans =

   -1.6000
    5.4000
    1.2000
```

inv

```
A=[2 1 -1;-3 -2 5;1 1 1];
b=[1 0 5]'
```

```
inv(A)*b

ans =

   -1.6000
    5.4000
    1.2000
```

Left devision

A=[2 1 -1;-3 -2 5;1 1 1];
b=[1 0 5]'

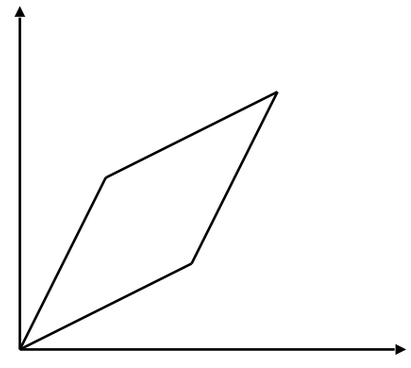
- A\b
- % Left devision
- % Ax=b could be solved by left devision

Determinant

```
A=reshape(1:4,2,2);
>> det(A)

ans =

    -2
```

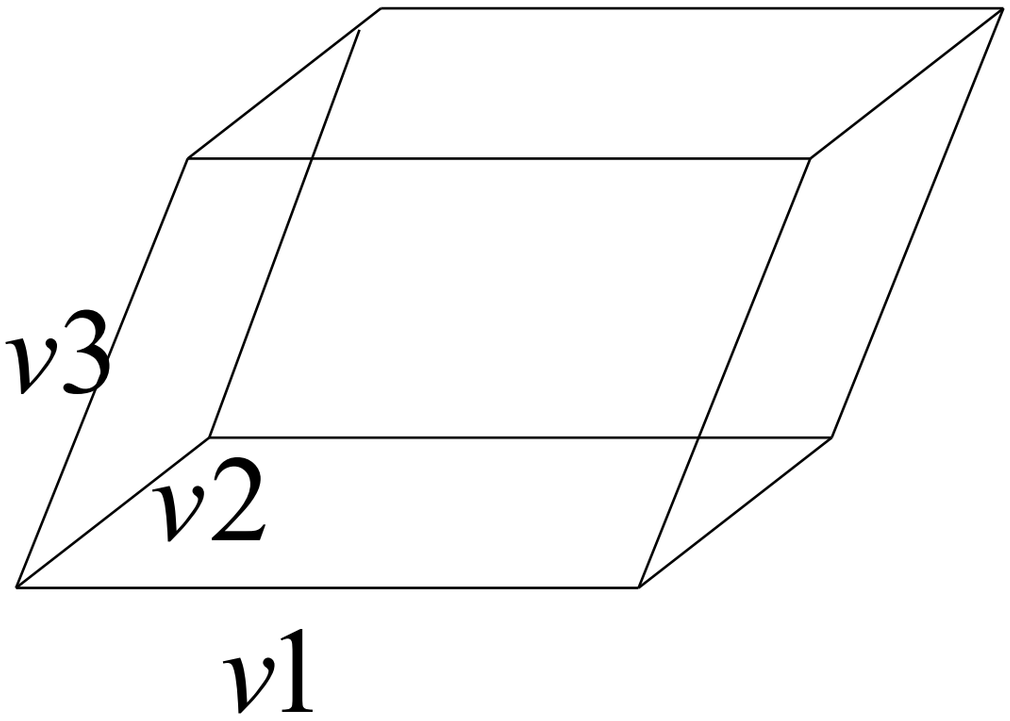
Area of a parallelogram

- v1,v2 denotes two vectors in a plane
- Calculate the area of the parallelogram determined by v1 and v2



Volume of a parallelepiped

A=[v1;v2;v3]
det(A)



```
reshape(randperm(9),3,3)

ans =

     5     9     6
     2     1     8
     7     3     4
```

Append Horizontally

```
>> A=reshape(randperm(9),3,3);
>> B=reshape(randperm(9),3,3);
>> C=[A B]
```

C =

6	2	9	5	4	9
3	4	5	6	3	7
8	7	1	8	1	2

Append Vertically

```
>> C=[A;B]

A =

     6     2     9
     3     4     5
     8     7     1
     5     4     9
     6     3     7
     8     1     2
```