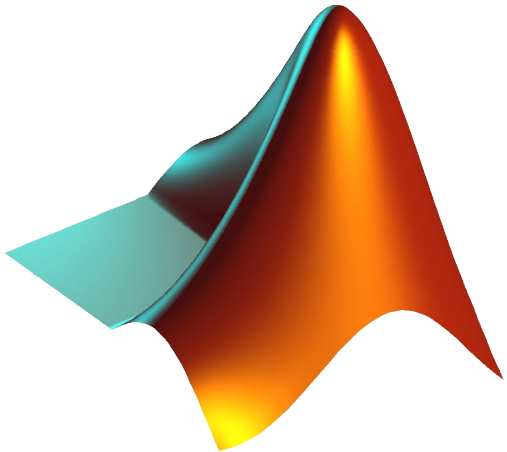
CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz



Website: https://www.cs.cornell.edu/courses/cs1112/2022fa/

Today: 2D arrays and images

Agenda and announcements

- Last time
  - 2D arrays
  - Triangular traversal
- Today
  - More 2D arrays
  - Working with images
- Announcements
  - Discussion 07 exercise due tonight 10/13 at 9 PM (check off and MATLAB grader)
  - Discussion 08 (yesterday) had optional problems – problems to help you study for prelim 1
  - Project 4 released tonight or tomorrow (will be due 10/26)
  - Prelim 1 is next Tuesday 10/18 from 7:30 - 9 in Klarman Hall (KG70). Students with approved exceptions please check CMS for your time/location.
  - Consultants will be holding tutoring (sign up on CMS)
    - Thurs (10/13), Sunday (10/16), and Monday (10/17)
  - Review session Sunday 10/16 from 6:30 - 8 PM in Philips 101

Extra notes about the prelim

- You do not need to write comments on the exam (you can if you want though)
- Any useful MATLAB built-in functions (like factorial, rand, ceil) that you might need on the exam will be listed on the front page of the exam
- You won't be marked down for most formatting things (for example, indenting in a for-loop or if-statement) but please still try to use good formatting

You may find the following MATLAB predefined functions useful: `abs`, `sqrt`, `rem`, `floor`, `ceil`, `rand`, `zeros`, `ones`, `linspace`, `length`, `input`, `fprintf`, `disp`

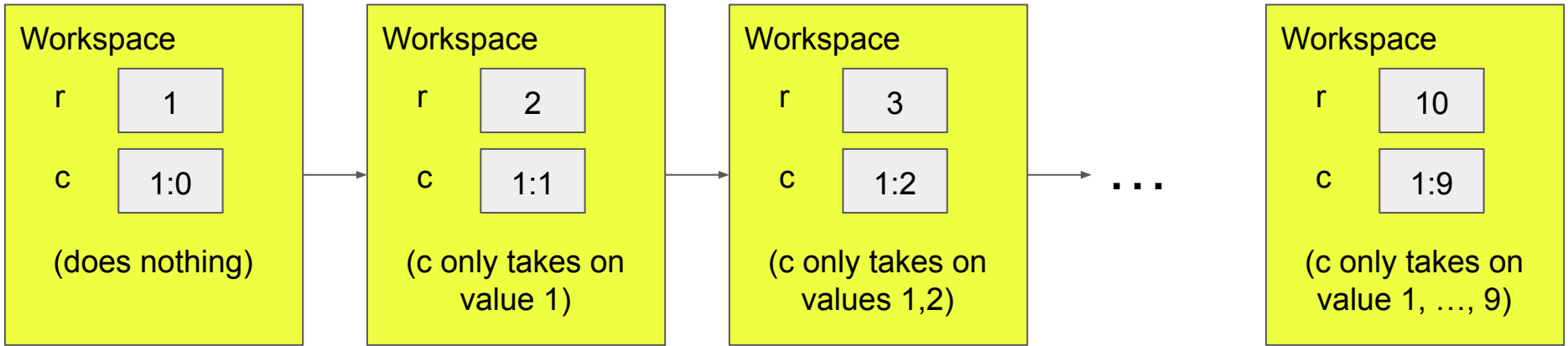
Examples:
`rem(5,2)` → 1, the remainder of 5 divided by 2
`rand()` → a random real value in the interval (0,1)
`abs(-3)` → 3, absolute value
`floor(6.9)`, `floor(6)` → 6, rounds down to the nearest integer
`ceil(8.1)`, `ceil(9)` → 9, rounds up to the nearest integer
`length([2 4 8])` → 3, length of a vector
`zeros(1,4)` → 1 row 4 columns of zeros
`linspace(3,5,10)` → a vector of 10 real numbers evenly distributed in the interval [3,5]

Triangular traversal: recap

```
% traverse the green cells
[nr, nc] = size(A);
for r = 1:nr
    for c = 1:r-1

        % Do something with A(r,c)

    end
end
```

A

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



Application of 2D arrays: images

Matrix of size 1080 x 1920

1080 x 1920 = 2,073,600 pixels

235	231	231	234	226	226	224	222	222	223
209	211	220	230	236	235	232	228	227	229
132	128	132	153	168	171	174	175	172	162
102	87	72	74	77	78	83	96	107	103
109	104	98	100	108	110	110	113	119	122
94	95	99	102	105	105	113	118	94	99

Images can be encoded in different ways

- Common formats include
  - JPG (or JPEG): better at compressing images
  - PNG: preserves all details but better for transparent backgrounds
- MATLAB (and many of the things we'll do) works well with many different image formats
- We'll work mostly with JPG files but the most important function will also work for PNG files
  - `imread`: read an image file and convert it to a numeric array they we can work with
  - `imshow`: display the numeric array as an image
  - `imwrite`: Write an array into an image file

Application of 2D arrays: images

Black corresponds to 0
White corresponds to 255

235	231	231	234	226	226	224	222	222	223
209	211	220	230	236	235	232	228	227	229
132	128	132	153	168	171	174	175	172	162
102	87	72	74	77	78	83	96	107	103
109	104	98	100	108	110	110	113	119	122
94	95	99	102	105	105	113	118	94	99

Each integer in this array corresponds to a single pixel and is of type `uint8`

(`uint8`: integer between 0 and 255)

Example: Let’s put a picture in a frame

Things to do:

1. Read (store) the image from your computer memory and convert it into an array with `imread`
2. Show original image with `imshow`
3. Assign a gray value to the edge pixels (edge values of the matrix)
4. Show the manipulated picture with `imshow`

Reading and showing an image

```
% read the image and convert it to a uint8 array img
img = imread('babyYoda_bw.jpg');

% show the image in a figure window
imshow(img)
```

Warning: the image must be in the same folder as the current script or else you will get an error message “File ‘babyYoda_bw.png’ does not exist.”

Code to frame a grayscale picture

```
img = imread('babyYoda_bw.jpg');
imshow(img)

% change the color of edge pixels









imshow(img)
```

Code to frame a grayscale picture

```
img = imread('babyYoda_bw.jpg');
imshow(img)

% change the color of edge pixels
width = 20;
frameColor = 200;   % light gray
[nr, nc] = size(img);


% loop through pixels and change pixel color if at border pixel




imshow(img)
```

width

Code to frame a grayscale picture

```
img = imread('babyYoda_bw.jpg');
imshow(img)

% change the color of edge pixels
width = 20;
frameColor = 200;   % light gray
[nr, nc] = size(img);
for r = 1:nr
    for c = 1:nc

        % change img(r,c) if we're at a border pixel

    end
end
imshow(img)
```

Code to frame a grayscale picture

```
img = imread('babyYoda_bw.jpg');
imshow(img)

% change the color of edge pixels
width = 20;
frameColor = 200;   % light gray
[nr, nc] = size(img);
for r = 1:nr
    for c = 1:nc
        if r <= width || r > nr-width || c <= width || c > nc-width
            img(r,c) = frameColor;
        end
    end
end
imshow(img)
```

Remember: img is a 2D array of uint8 integers but frameColor is of type double. Is this fine?

Yes! MATLAB will automatically do the conversion from double to uint8!

Code to frame a grayscale picture

```
img = imread('babyYoda_bw.jpg');
imshow(img)

% change the color of edge pixels
width = 20;
frameColor = 200;   % light gray
[nr, nc] = size(img);
for r = 1:nr
    for c = 1:nc
        if r <= width || r > nr-width || c <= width || c > nc-width
            img(r,c) = frameColor;
        end
    end
end
imshow(img)
```

Accessing a submatrix

M

0	1	5	9	7
7	-5	-1	20	26
19	-8	13	4	2

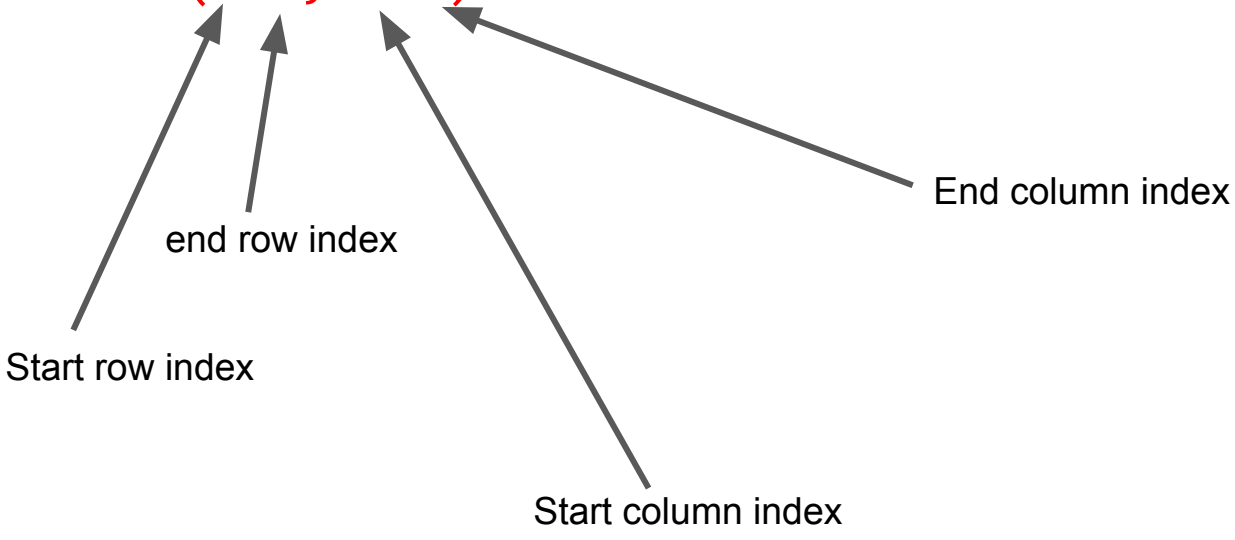
- `M` refers to the whole matrix
- `M(3,5)` refers to the element in the 3rd row and 5th column of `M`

Accessing a submatrix

M

0	1	5	9	7
7	-5	-1	20	26
19	-8	13	4	2

- `M` refers to the whole matrix
- `M(3,5)` refers to the element in the 3rd row and 5th column of `M`
- `M(2:3, 1:4)` refers to a submatrix of `M`



Changing all values in a submatrix

end indicates that I want to the last row (the end)

```
M(2:end, 1:4) = 99*ones(2,4);

% easier syntax that works the same!
M(2:end, 1:4) = 99;
```

M

0	1	5	9	7
7	-5	-1	20	26
19	-8	13	4	2

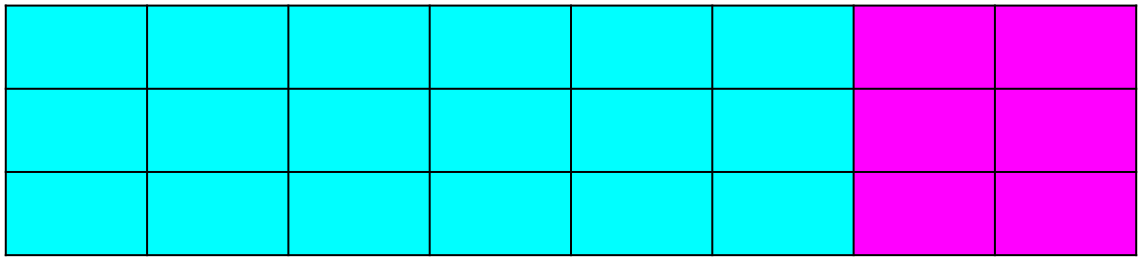
M

0	1	5	9	7
99	99	99	99	26
99	99	99	99	2

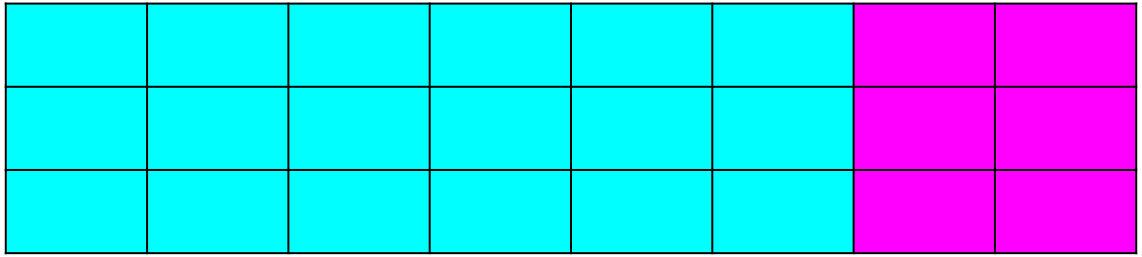
More efficient code to frame an image

```
img = imread('babyYoda_bw.jpg');
imshow(img)

% change the color of edge pixels
width = 20;
frameColor = 200;   % light gray
[nr, nc] = size(img);

img(1:width,:) = frameColor; % change top rows
% add code here to deal with bottom, left, and right borders


imshow(img)
```

How can we change the last n columns of a 2D array to 5?



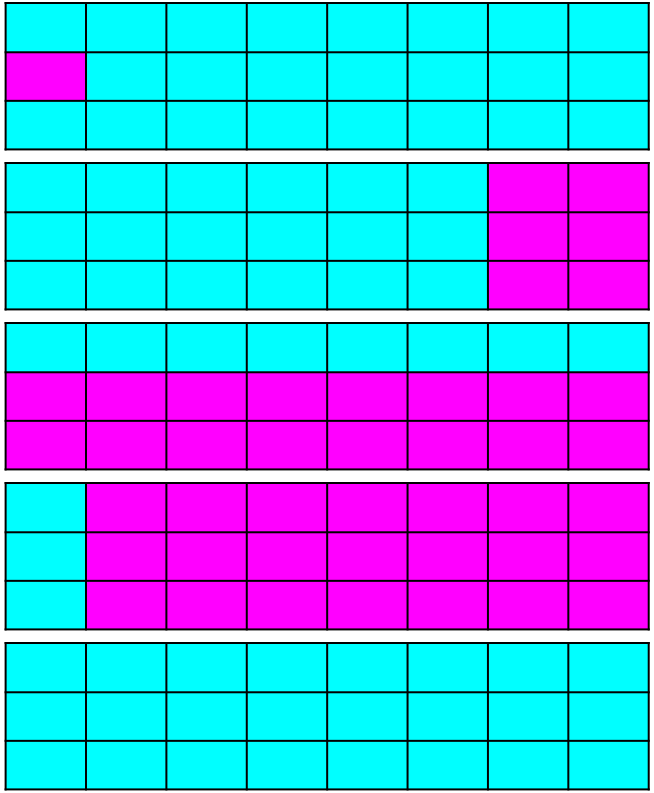
n = 2



n = 2
nr = 3
nc = 8

```
A( n ) = 5;

A( : , nc-n+1:end ) = 5;

A( n:end , : ) = 5;

A( : , n:end ) = 5;

A( nc-n+1:end , : )
```

7



Color images

A color image is made up of RGB matrices → 3D array!

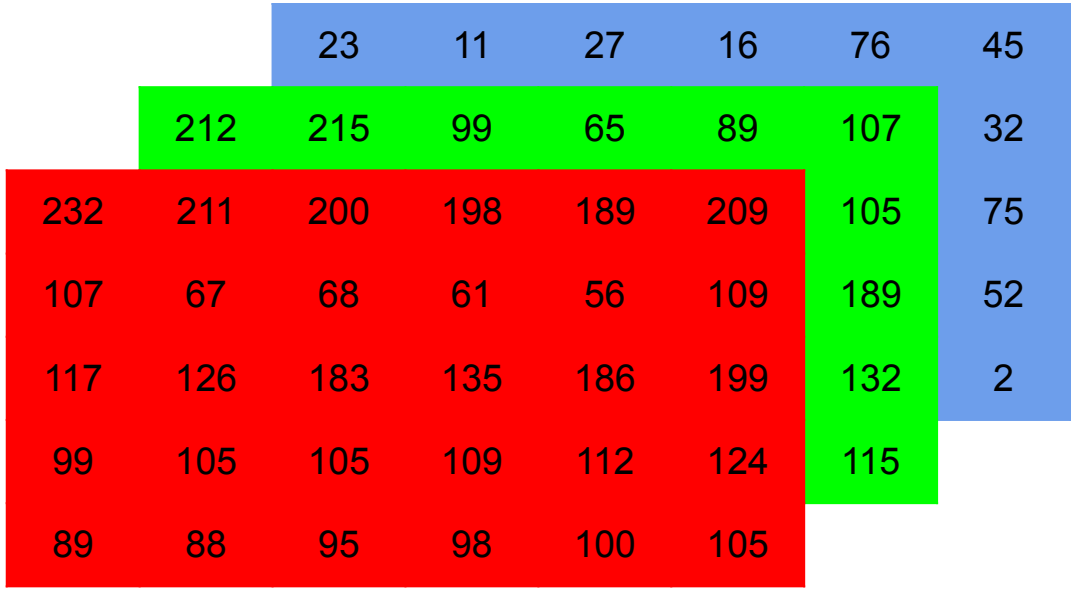
Now we need 3 indices to represent elements:

```
img_colr(r,c,l)
    r: row
    c: column
    l: layer
```

```
For color images,
There are 3 layers (R, G, B)!
```

for `uint8` images,

```
0 <= img_colr(r,c,l) <= 255
```



Throwing errors: say we are writing a code that only works for grayscale images

```
img = imread('babyYoda.jpg');
[nr, nc, nl] = size(img);   % stores #rows, #cols, #layers
% nl = 3 for color image, nl = 1 for grayscale image

if nl == 3
    error('The image you are processing is color, not grayscale.')
end

% do something with grayscale image
```

The error function “throws an error” whenever it is evaluated. (The code stops and the error message is printed to the command line)