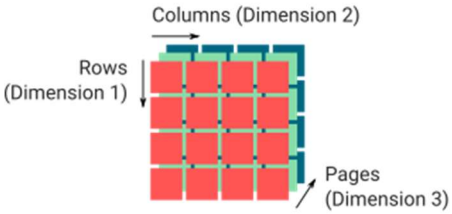
Quantitative Optical Imaging (MCDB/BCHM 4312/5312) 2019

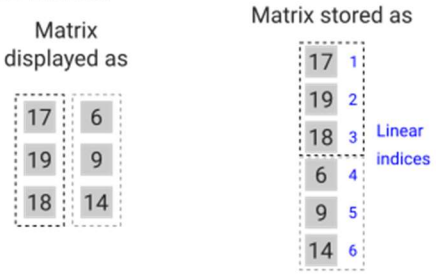
MATLAB Reference Sheet

Matrix operations

Matrix Dimensions



Linear Indices



Addition/Subtraction with a scalar

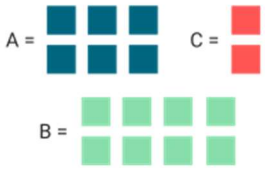
Addition/Subtraction with matrices

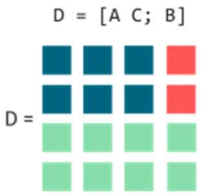
Not allowed if sizes are unequal

Matrix Creation

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

Concatenation

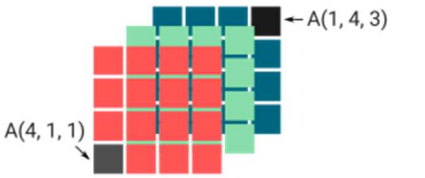




3D matrices: `cat(3, <matrix1>, <matrix2>, [...])`

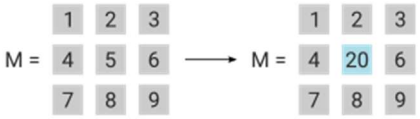
Subscript Indices

```
M(<row>, <column>, [<page>, ...])
```



Modifying elements

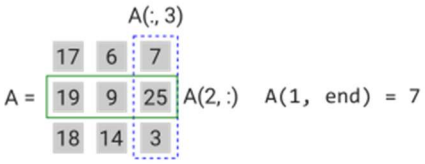
```
M(2, 2) = 20
```



Matrix multiplication (*)

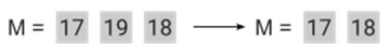
C(1, 1) = 1 * 4 + 2 * 2 = 8

Colon operator and end

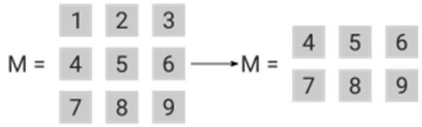


Deleting elements

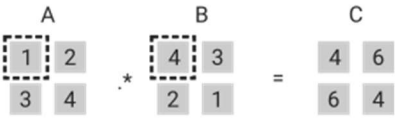
```
M(2) = []
```




```
M(1,:) = []
```

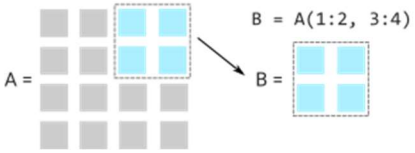


Element-wise multiplication (.*)

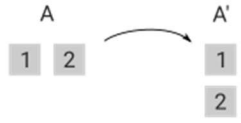


C(1,1) = 1 * 4

Extracting submatrices



Matrix Transposition (')



Structured arrays

Creating a struct array

```
S(1).Centroid = [1 2];
S(2).Centroid = [3 4];
```

Indexing data from a struct array

```
pos1 = S(1).Centroid;
x2 = S(2).Centroid(1);
```

List of functions
<> indicate optional arguments, 'str' indicate strings

Documentation
`help function_name`
Print help for specified function in Command Window
`doc function_name`
Display HTML help for specified function

Matrices
`double(M)`
Convert elements of matrix M to double-precision format
`uint8(M); uint16(M)`
Convert elements of matrix M to unsigned 8-bit or 16-bit integer format
`linspace(start, end, N)`
Generates a vector with N elements, including both start and end points
`start:<step:>end`
Generates a vector starting at the start point, with an optional step size (default 1). Might not include `end` value if end point cannot be reached by the step size.
`size(M<, dim>)`
Returns the size of the matrix (along the dimension specified)
`numel(M)`
Returns number of elements in M
`nnz(L)`
Returns number of true elements in L
`cat(dim, M1, M2, …)`
Joins (concatenate) matrices along dimension specified
`[X, Y] = meshgrid(x, y)`
Calculates the grid matrices from vectors x and y
`NaN<(numrows, numcols)>`
Create a matrix of specified size filled with NaNs (Not-a-Number)
`isnan(M)`
Returns true for elements that are NaN
`zeros(numrows, numcols)`
Creates a matrix of zeros
any(L<, dim>)
Returns true if any element of the logical array L is true
all(L<, dim>)
Returns true if every element of the logical array L is true

Statistics and trigonometry
`min(M, <[], dim>); min(M, [], 'all')`
Compute minimum elements of an array along the specified dimension or over all elements
`max(M, <[], dim>); max(M, [], 'all')`
Compute maximum elements of an array along the specified dimension or over all elements
`mean(M<, dim>); mean(M, 'all')`
Compute mean of an array along the specified dimension or over all elements
`median(M<, dim>); median(M, 'all')`
Compute median of an array along the specified dimension or over all elements
`cos(M); cosd(M)`
Compute cosine of each element in radians or degrees
`sin(M); sind(M)`
Compute cosine of each element in radians or degrees
`acos(M); acosd(M)`
Compute inverse cosine of each element (M in radians or degrees)
`asin(M); asind(M)`
Compute inverse sin of each element (M in radians or degrees)
`tan(M); tand(M)`
Compute tangent of each element (M in radians or degrees)
`atan(M); atand(M)`
Compute inverse tangent of each element (M in radians or degrees)
`atan2(Y, X); atan2d(Y, X)`
Compute inverse tangent of each element (M in radians or degrees), correcting sign using the quadrant of the specified coordinate

Plotting
`figure`
Create a new figure window
`plot(x, y<, linespec>)`
Plots elements in x vs y
`scatter(x, y)`
Scatter plot of x vs y
`boxplot(x<, g>)`
Creates a box plot of data in x (optional g is the group)
`histogram(M<, N>)`
`histogram(M, Option, Value)`
Plot histogram with optional N number of bins or other options (such as `'BinWidth'`)
`semilogx(x, y<, linespec>)`
Plots x-axis on a log scale
`semilogy(x, y<, linespec>)`
Plots y-axis on a log scale
`loglog(x, y<, linespec>)`
Plots both axes on log scales
`hold on; hold off`
Hold on keeps previous plot (e.g. to overlay multiple line plots)
`legend('str1', 'str2', …)`
Draws legend, one legend string per line in plot
`xlabel('string')`
`ylabel('string')`
Label axes
`title`
Titles the plot
`xlim([min max]); ylim([min max])`
Sets limits on the the x- (or y-) axis

Curve-fitting
`fitObj = fit(x, y, 'model')`
Fit curve (1D) to the model specified
`plot(fitObj, x, y)`
Plots original data and fitted model
`fit([x, y], z, 'model')`
`fit([x, y], z, fitType)`
Fit surface (2D) to the model specified
`fittype('expression', name, value)`
Specifies a fit type object (custom model for fitting)

Images
`imfinfo('filename')`
Get image file information. Number of elements is number of pages of a multi-page TIFF
`imread('filename', <frame>)`
Read a single image from file (optional: frame number)
`imshow(I<, [min, max]>`
`imshow(I, [])`
Display image
`imshowpair(A, B)`
Displays a composite of two images with A in green, B in magenta
`imwrite(I, 'filename')`
Write matrix I to image file
`colorbar`
Display colorbar on current image
`rgb2lab(I)`
Converts an RGB image to L*a*b* color space

Image analysis

`regionprops(M<, I><, 'Property'>)`
Measure data from masks (e.g. centroid positions, mean intensity)

```
[centers, radii] = ...
    imfindcircles(I, [min_radius, max_radius])
```

Find circles of specified radius in image I

`viscircles(centers, radii)`
Draws circles from `imfindcircles`

`imdistline`
Interactive line distance tool (use after displaying image)

`improfile`
Measure intensity profile interactively (use after displaying image)

`imbinarize(I)`
Generates a mask by intensity thresholding using Otsu's method

`imfill(M, 'holes')`
Fills in holes (zeros surrounded by ones) in mask

`imclearborder(M)`
Clear connected true pixels touching image edge

`bwperim(M)`
Returns a logical array containing the perimeter of the mask M

`bwareaopen(L, minarea)`
Removes true regions less than minarea pixels

`strel('disk', radius)`
`strel('line', length, angle)`
Generates structuring elements

`imerode(I, SE)`
Morphological erosion on image I with structuring element SE

`imdilate(I, SE)`
Morphological dilation on image I with structuring element SE

`imopen(I, SE)`
Morphological opening on image I with structuring element SE

`imclose(I, SE)`
Morphological closing on image I with structuring element SE

`bwdist(L)`
Compute the distance transform of the mask L

`watershed(I)`
Runs the watershed algorithm on the image I

`imhmin(dd, h)`
Suppresses minima with depth less than h

`imgaussfilt(I, sigma)`
Computes the Gaussian blur filter on the image with a standard deviation of sigma

`medfilt2(I)`
Computes the median filter of the image I

`imregionalmax(I)`
Returns a logical array with positions of regional maxima

If statement

- Useful for controlling flow of code depending on value of variables

```
if logical_expression
  %Statement body executes if the logical_expression
  %is true. If not, check the next statement.
  %logical_expression should be a statement such as
  %var > 5 or any(A > 5)

elseif logical_expression
  %You can have multiple elseif statements.

else
  %Statement body executes if none of the
  %logical_expression statements were true.
end
```

Logical operators:
`<, <=, >, >=, ==, ~=, ~`

For loop

- Useful for repeating commands a set number of times
- `ind` = index variable, value changes to the next element in values each time the loop is run
- `values` = index values, the loop runs `N = numel(values)` times

```
for ind = values
    %Statement body executes each loop
    %The value of ind changes each time the loop is run:
    %  First loop, ind = values(1)
    %  Second loop, ind = values(2) ...
    %  N-th loop, ind = values(N)
end
```

Examples:

Storing data in a for loop
Basic steps:

- Initialize a matrix to store data, ideally with the correct size or with a size much larger than the expected number of elements. This step is important because growing a matrix is very time and memory intensive
- Index the storage element(s) in the for loop and assign them to the value of the data
- Remove matrix values which were unassigned

Example:

```
%Initialize a matrix of NaNs to store data
storeData = nan(100, 2);

for idx = 1:50
    %Do things in the loop, e.g. load in next image,
    %segment objects, measure data

    %Store the data (this example assumes you're storing
    %a 1x2 vector, e.g. for centroid coordinates)
    storeData(idx, :) = data.Centroid;
end

%Remove the unassigned matrix rows (these rows consist of
%NaNs)
rowsWithNaNs = any(isnan(storeData), 2);
storeData(rowsWithNaNs, :) = [];
```

Reading frames from a multi-page TIFF

- Remember that `imread` can only one image in at a time, so to process all frames, you must read them in one at a time using a for loop

```
%Get number of frames using imfinfo (returns a struct
%array with one element per frame)
numFrames = numel(imfinfo('imagefile.tif'));

for currFrame = 1:numFrames
    %Read in the next frame, using the index variable of
    %the for loop to keep track of the current frame
    %number
    currImage = imread('imagefile.tif', currFrame);

    %Analyze the image
end
```