Linear Algebra

Matrices Part 1:
Matrices and Basic Operations

Automotive Intelligence Lab.

Contents

Matrix: introduction

Introduction of Matrix

Matrix: Vector taken to the next level

- Highly versatile mathematical objects.
  - Equations
  - Geometric transformations
  - Positions of particles over time
  - Financial records
  - Myriad other things

- We can use matrix in data science either.
  - Rows: Observations (e.g., customers)
  - Columns: Features (e.g., purchases)

Visualizing

- **Small matrices can simply be printed out in full.**
  - But matrices that you work with in practice can be large.
    - Larger matrices will be visualized as $\boxed{\text{images}}$
    - Numerical value of each element of the matrix maps onto a color in the image.
    - Maps are pseudo-colored.
      - Mapping of numerical value onto color is arbitrary.

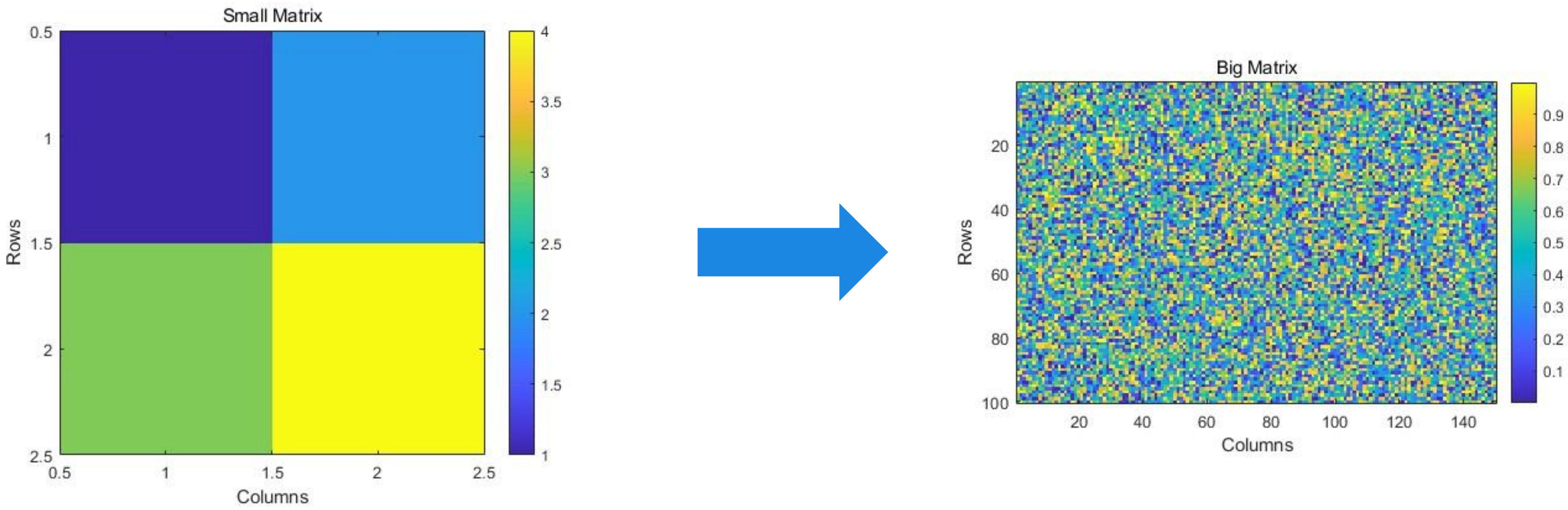


Image of small and big matrix

Indexing and Slicing

- **Matrices are indicated using bold-faced capital letters, like matrix $\boldsymbol{A}$ or $\boldsymbol{M}$.**
  - Size of a matrix is indicated using (row, column) convention.

- **Refer to specific elements of a matrix.**
  - Indexing the row and column position.
  - Element in the 3rd row and 4th column of matrix $\boldsymbol{A}$ is indicated as $a_{3,4} = \boxed{8}$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 0 & 2 & 4 & 6 & 8 \\ 1 & 4 & 7 & 8 & 9 \end{bmatrix}$$

Example of matrix

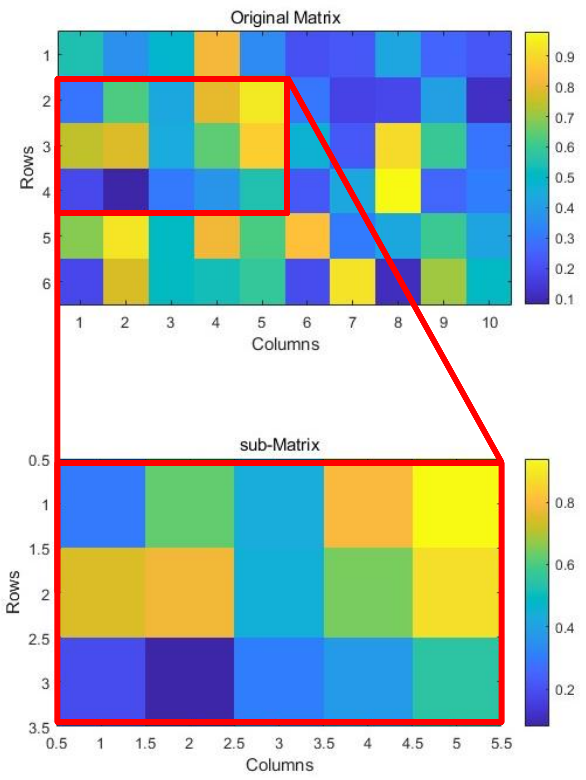
- **Extracting a subset of rows or columns of a matrix is done through slicing.**
  - Following code shows an example of extracting a submatrix from rows 2-4 and columns 1-5 of a large matrix.

Code Exercise of Indexing and Slicing

- **Extracting a subset of rows or columns of a matrix is done through slicing.**
  - Remember that **MATLAB** uses **index start from 1**, not 0 like others.

Code Exercise (05_01)

```
% Clear previous figures and vars
clc; clear; close all;

% Create a 6x10 matrix with random values
originalMatrix = rand(6, 10);

% Extract submatrix from rows 2 to 4 and columns 1 to 5
subMatrix = originalMatrix(2:4, 1:5);

% Display
disp("Original Matrix");
disp(originalMatrix);
disp("Sub-Matrix");
disp(subMatrix);

% Visualize the original matrix
figure; % Create a new figure for the original matrix
imagesc(originalMatrix); % Display the original matrix as a color image
title('Original Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the data

% Visualize the sub-matrix
figure; % Create a new figure for the sub-matrix
imagesc(subMatrix); % Display the sub-matrix as a color image
title('sub-Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the data
```



MATLAB code example of slicing matrices and results

Special Matrices

- **Number of matrices is infinite.**
  - Infinite number of ways of organizing numbers into a matrix.

  - But matrices can be described using a relatively small number of characteristics.
    - It creates “families” or categories of matrices.
    - These categories appear in certain operations.
      - They have certain useful properties.
      - Some categories of matrices are used so frequently that they have dedicated MATLAB functions to create them.

Special Matrices: Random Numbers

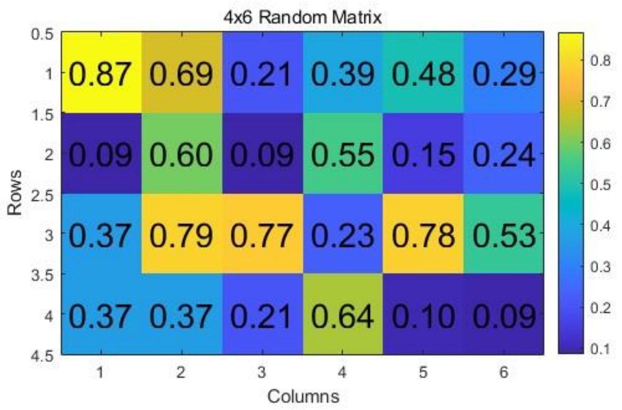
- **Matrix can contain numbers drawn at random from some distribution, typically $\boxed{\text{Gaussian}}$ (a.k.a. normal).**
  - Random-numbers matrices
    - Can be quickly and easily created with any size and rank.

Special Matrices: Random Numbers

- **Several ways to create random matrices in MATLAB.**
  - Random matrices with **float** numbers using function ‘rand(row,Col)’.
  - Random matrices with **integer** numbers using function ‘rani(row,col)’.

Code Exercise (05_02)

```
% Clear previous figures and vars
clc; clear; close all;

% 4x6 matrix with random values
matrix = rand(4, 6);

% Create a figure for visualization
figure;

% Display
disp("4x6 Random Matrix")
disp(matrix);

% Visualize the matrix
imagesc(matrix); % Display the matrix as a color image
title('4x6 Random Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the data
[numRows, numCols] = size(matrix);
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(matrix(row, col), '%0.2f'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end
```

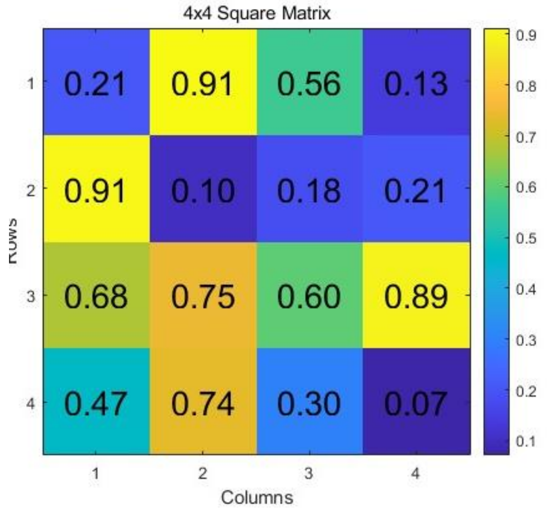


MATLAB code to make random numbers matrix and result

Special Matrices: Square VS Non-square

Square matrix

- Same number of rows as columns.
- Can be expressed as $R^{N \times N}$.



Square matrix

Non-square matrix

- Different number of rows and columns.
  - Sometimes called a rectangular matrix.
- Can be called tall.
  - Number of rows $>$ number of columns
- Can be called wide.
  - Number of rows $<$ number of columns

You can create square and rectangular matrices from random numbers.

- By adjusting the shape parameters in the previous code.

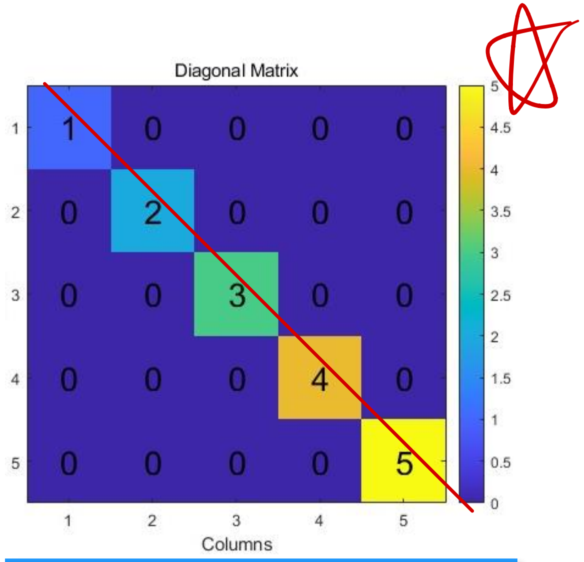
Special Matrices: Diagonal

Diagonal of a matrix

- Elements starting at the $\boxed{\text{top-left}}$ and going down to the $\boxed{\text{bottom-right}}$

Diagonal matrix

- Zeros on all the off-diagonal elements.
- Diagonal elements are the only elements that may contain nonzero values.
  - Diagonal elements can contain zero too.



Diagonal Matrix

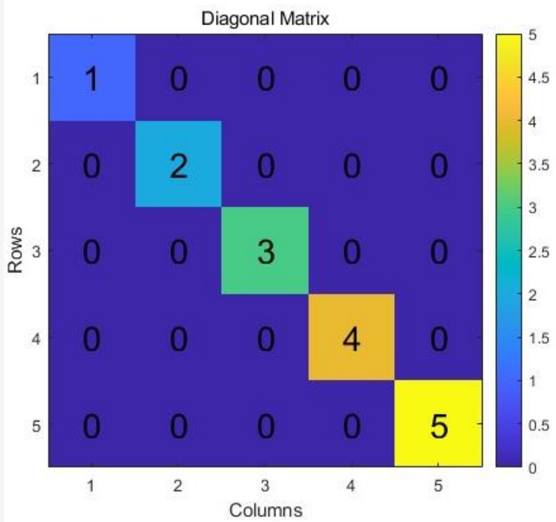
Code Exercise of Diagonal Matrix

MATLAB function diag()

- Input as matrix: diag() will return the diagonal elements of matrix as a vector.
- Input as vector: diag() will return a matrix with input vector elements on the diagonal.

Code Exercise (05_03)

```
% Clear previous figures and vars
clc; clear; close all;

% Create a vector for the diagonal
elements
InputMatrix = randi(10, 5);
diagonalElements = [1, 2, 3, 4, 5];

% Create a diagonal matrix using the diag
function
diagonalVector = diag(InputMatrix);
diagonalMatrix = diag(diagonalElements);

% Display
disp("Input Matrix");
disp(InputMatrix);
disp("Diagonal Vector")
disp(diagonalVector);

% Create a figure for visualization
figure;

% Visualize the diagonal matrix
imagesc(diagonalMatrix); % Display the
matrix as a color image
title('Diagonal Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit
tightly around the data
set(gca, 'XTick',
1:length(diagonalElements), 'YTick',
1:length(diagonalElements)); % Set the
tick marks

% Add text annotations for each element
[numRows, numCols] = size(diagonalMatrix);
for row = 1:numRows
    for col = 1:numCols
        text(col, row,
num2str(diagonalMatrix(row, col),
'%d'), ...
            'HorizontalAlignment',
'center', ...
            'VerticalAlignment',
'middle', ...
            'FontSize',20);
    end
end
```




```
Input Matrix
     8     9     5     1     7
     4     9     4     6     1
     6     3     2     5     1
     2     6     2     7     4
    10     1     5     7     6

Diagonal Vector
     8
     9
     2
     7
     6
```

MATLAB code of function 'diag()' and results

Special Matrices: Triangular

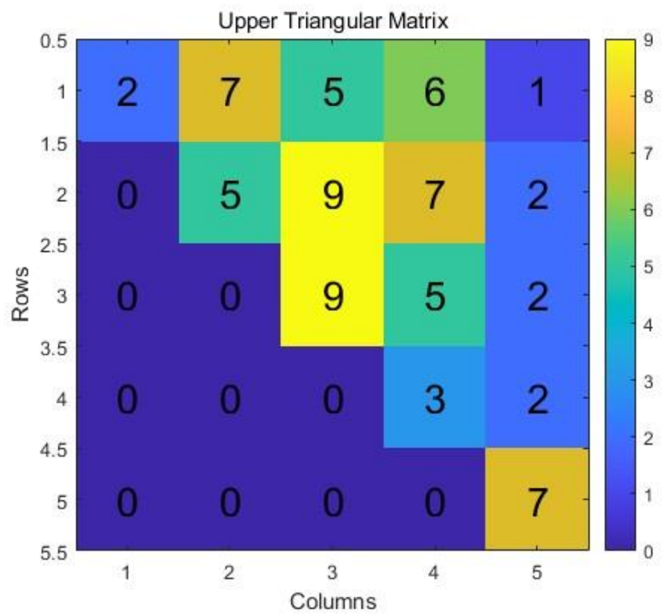
- **Triangular matrix**
  - Contains all zeros either above or below the main diagonal.
  - Upper triangular
    - if the nonzero elements are above the diagonal.
  - Lower triangular
    - if the nonzero elements are below the diagonal.

Code Exercise of Triangular Matrix

Code Exercise (05_04)

- Create upper triangular matrix using function ‘triu(matrix)’
- Create lower triangular matrix using function ‘tril(matrix)’.

```
% Clear workspace, command window, and
close all figures
clc; clear; close all;

% Create a random matrix using randi
randomMatrix = randi(10, 5, 5); % Generate
a 5x5 matrix with integers between 1 and
10

% Create upper triangular matrix from the
random matrix
upperTriangularMatrix = triu(randomMatrix);

% You can create lower triangular matrix
use function 'tril(matrix)'

% Create and visualize the upper
triangular matrix
figure;
imagesc(upperTriangularMatrix); % Display
the matrix as a color image
title('Upper Triangular Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit
tightly around the data
% Add text annotations for each element in
the upper triangular matrix
[numRows, numCols] =
size(upperTriangularMatrix);
for row = 1:numRows
    for col = 1:numCols
        text(col, row,
num2str(upperTriangularMatrix(row, col),
'%d'), ...
            'HorizontalAlignment',
'center', ...
            'VerticalAlignment',
'middle', ...
            'FontSize',20);
    end
end
```



MATLAB code to create triangular matrix and result

Special Matrices: Identity

Identity matrix

- Equivalent of the number $\boxed{1}$
  - In that any matrix or vector times the identity matrix is that same matrix or vector.

- Equivalent of Square diagonal matrix
  - With all diagonal elements having a value of $\boxed{1}$.

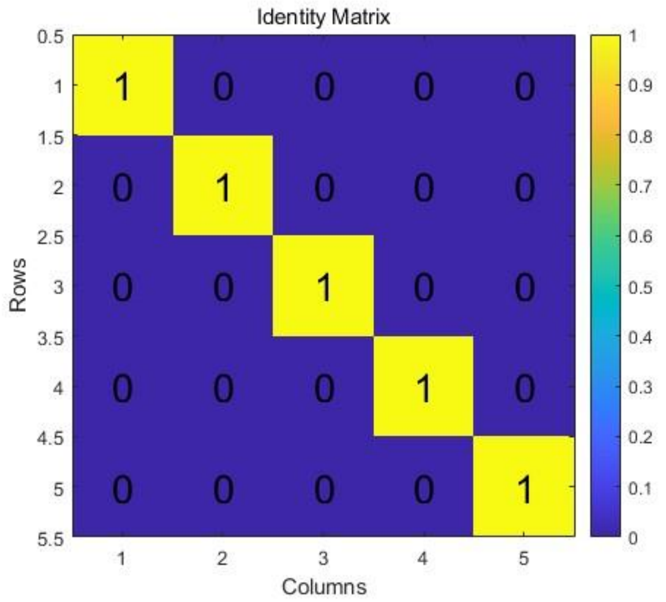
Notation of Identity matrix

- Indicate using the letter $\mathbf{I}$.
  - I_5 is 5×5 identity matrix.
  - If there is no subscript to indicate its size, you can infer the size from context.
    - To make the equation consistent.

Code Exercise of Identity Matrix

- **Identity matrix can be created by eye() in MATLAB code.**
- **Code Exercise (05_05)**

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Create a 5x5 identity matrix
identityMatrix = eye(5, 5);

% Create and visualize the identity matrix
figure;
imagesc(identityMatrix); % Display the matrix as a color image
title('Identity Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the data
% Add text annotations for each element in the identity matrix
[numRows, numCols] = size(identityMatrix);
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(identityMatrix(row, col), '%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end
```



MATLAB code to create identity matrix and result

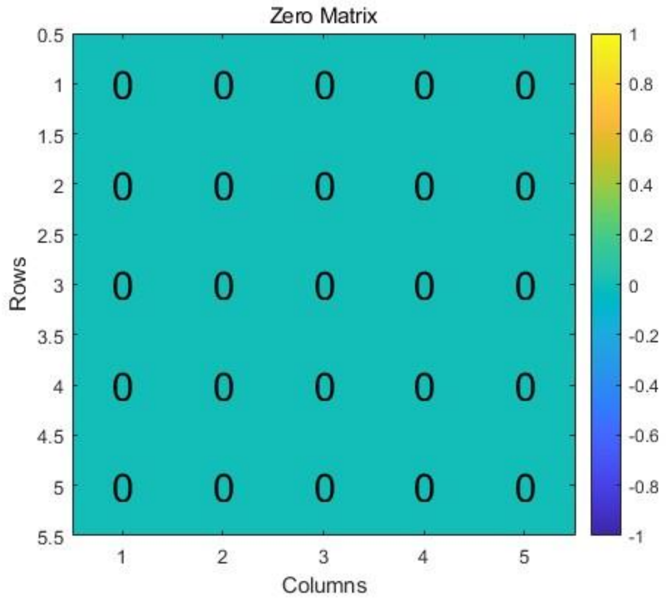
Special Matrices: Zeros

- **All of zeros matrix elements are zero.**
  - Indicate using a bold-faced zero: $\mathbf{0}$.
  - It can be a bit confusing to have the same symbol indicate both a vector and a matrix.
    - But this kind of overloading is common in math and science notation.

Code Exercise of Zero Matrix

- Using the zeros() function in MATLAB to create zeros matrix.
- Code Exercise (05_06)

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Create a 5x5 zero matrix
zeroMatrix = zeros(5, 5);

% Create and visualize the zero matrix
figure;
imagesc(zeroMatrix); % Display the matrix as a color image
title('Zero Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the data
% Add text annotations for each element in the zero matrix
[numRows, numCols] = size(zeroMatrix);
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(zeroMatrix(row, col), '%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end
```



MATLAB code to create zero matrix

Matrix math: addition, scalar multiplication, Hadamard multiplication

Matrix Math: Addition and Subtraction

Matrix addition

- Defined only between two matrices of the $\boxed{\text{Same size.}}$.

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 3 & 1 \\ -1 & -4 & 2 \end{bmatrix} = \begin{bmatrix} (2+0) & (3+3) & (4+1) \\ (1-1) & (2-4) & (4+2) \end{bmatrix} = \begin{bmatrix} 2 & 6 & 5 \\ 0 & -2 & 6 \end{bmatrix}$$

Matrix addition example

Matrix Math: Shifting a Matrix

- **Linear-algebra is way to add a scalar to a square matrix.**
  - Called $\boxed{\text{Shifting a matrix}}$
  - Not formally possible to add a scalar to a matrix, as in $\lambda + \boldsymbol{A}$.
    - It works by adding a scalar multiplied identity matrix, as in $\mathbf{A} + \lambda \boldsymbol{I}$.

$$\begin{bmatrix} 4 & 5 & 1 \\ 0 & 1 & 11 \\ 4 & 9 & 7 \end{bmatrix} + 6 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 5 & 1 \\ 0 & 7 & 11 \\ 4 & 9 & 13 \end{bmatrix}$$

Matrix addition example

  - Only the diagonal elements can be changed.
    - The rest of the diagonal elements is unchanged by shifting.
  - In practice, one shifts a relatively small amount.
    - To preserve as much information as possible in the matrix while benefiting from the effects of shifting, including increasing the numerical stability of the matrix.

Matrix Math: Applications of Shifting a Matrix

- **Exactly how much to shift is a matter of on-going research in multiple areas.**
  - Such as machine learning, statistics, deep learning, control engineering, etc.

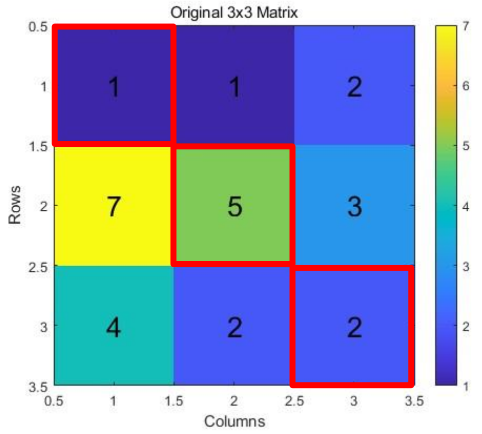
  - For example, is shifting by $\lambda = 6$ a little or a lot ? How about $\lambda = 0.001$?
    - These numbers are "big" or "small" relative to the numerical values in the matrix.
    - Therefore, in practice, λ is usually set to be some fraction of a matrix-defined quantity such as the norm or the average of the eigenvalues.
      - You will get to explore about norm and eigenvalues in later chapters.

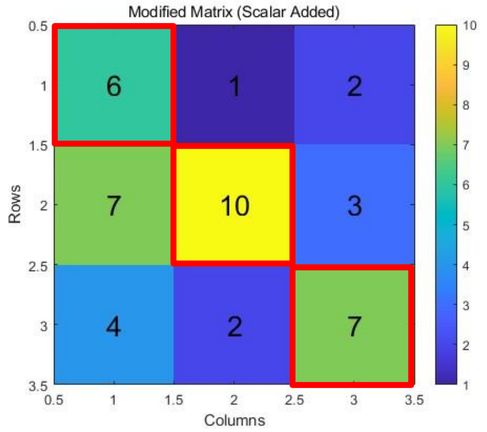
- **Two primary applications of shifting a matrix**
  - Finding the eigenvalues of a matrix.
  - Regularizing matrices when fitting models to data.

Code Exercise of Matrix Shifting

Code Exercise (05_07)

```
% Clear workspace, command window, and close all
figures
clc; clear; close all;

% Create a 3x3 square matrix with random values
squareMatrix = randi(10, 3, 3); % Generate a 3x3
matrix with integers between 1 and 10

% Define a scalar value
scalar = 5;

% Create a 3x3 identity matrix
identityMatrix = eye(3);

% Add the scalar multiplied by the identity matrix
to the original matrix
modifiedMatrix = squareMatrix + scalar *
identityMatrix;

% Create and visualize the original square matrix
in a new figure
figure;
imagesc(squareMatrix); % Display the matrix as a
color image
title('Original 3x3 Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly
around the data
% Add text annotations for each element in the
original matrix
[numRows, numCols] = size(squareMatrix);
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(squareMatrix(row,
col), '%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end

% Create and visualize the modified matrix in a
new figure
figure;
imagesc(modifiedMatrix); % Display the modified
matrix as a color image
title('Modified Matrix (Scalar Added)');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly
around the data
% Add text annotations for each element in the
modified matrix
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(modifiedMatrix(row,
col), '%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end
```





MATLAB code to calculate the matrix shifting

Matrix Math: Scalar-Matrix and Hadamard Multiplications

Scalar-matrix multiplication and Hadamard multiplication

- Work the same for matrices as they do for vectors, which is to say, element-wise.
- Scalar-matrix multiplication.
  - Multiply each element in the matrix by the same scalar.

$$\Upsilon \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \Upsilon a & \Upsilon b \\ \Upsilon c & \Upsilon d \end{bmatrix}$$

Example of scalar multiplication

- Hadamard multeiplication
  - Involving multiplying two matrices element-wise.

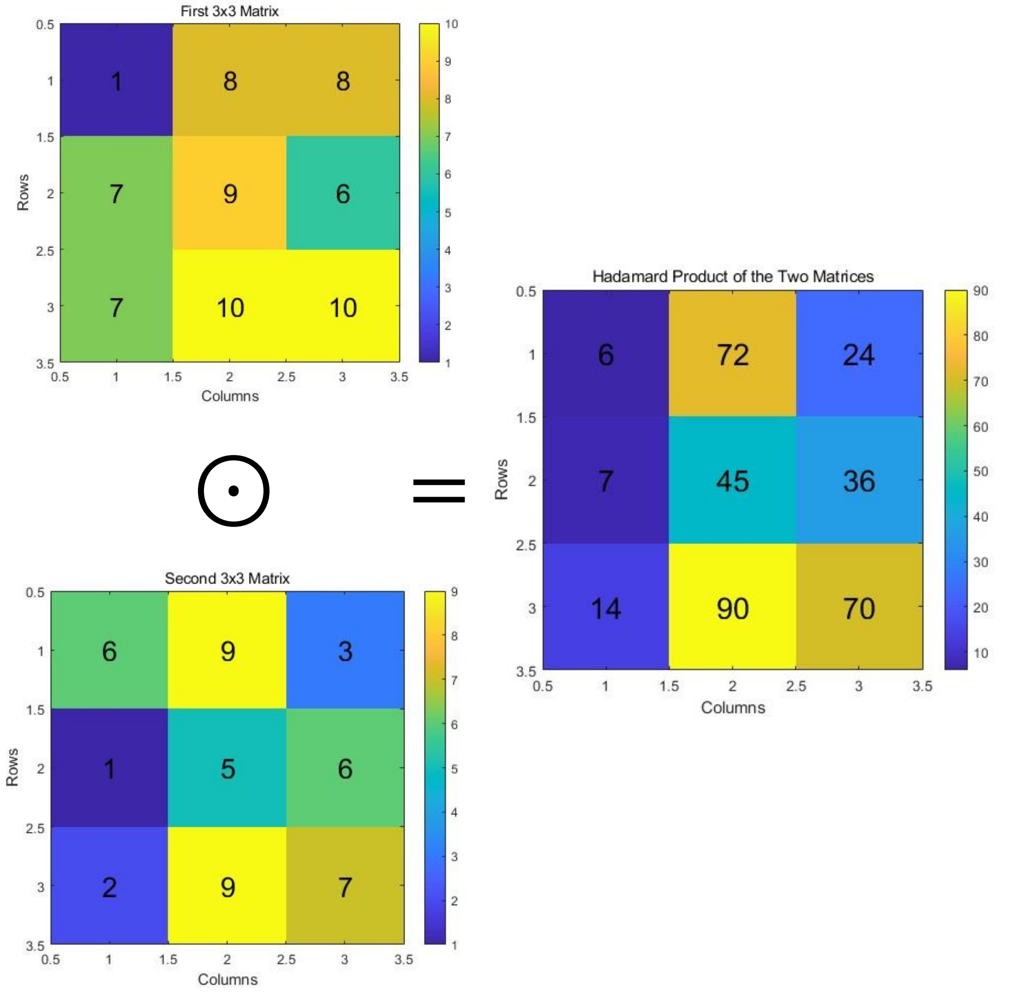
$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \odot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 2a & 3b \\ 4c & 5d \end{bmatrix}$$

Example of Hadamard multiplication

Code Exercise of Hadamard Multiplication

Code Exercise (05_08)

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Create two 3x3 square matrices with random values
squareMatrix1 = randi(10, 3, 3); % Generate a 3x3 matrix
with integers between 1 and 10
squareMatrix2 = randi(10, 3, 3); % Generate another 3x3
matrix with integers between 1 and 10

% Calculate the Hadamard product of the two matrices
hadamardProduct = squareMatrix1 .* squareMatrix2;

% Create and visualize the first square matrix
figure;
imagesc(squareMatrix1); % Display the matrix as a color
image
title('First 3x3 Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the
data
% Add text annotations for each element in the first
matrix
[numRows, numCols] = size(squareMatrix1);
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(squareMatrix1(row, col),
'%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end

% Create and visualize the second square matrix
figure;
imagesc(squareMatrix2); % Display the matrix as a color
image
title('Second 3x3 Matrix');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the
data
% Add text annotations for each element in the second
matrix
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(squareMatrix2(row, col),
'%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end

% Create and visualize the Hadamard product of the
matrices
figure;
imagesc(hadamardProduct); % Display the Hadamard product
matrix as a color image
title('Hadamard Product of the Two Matrices');
xlabel('Columns');
ylabel('Rows');
colorbar; % Show a color scale
axis equal tight; % Adjust axes to fit tightly around the
data
% Add text annotations for each element in the Hadamard
product matrix
for row = 1:numRows
    for col = 1:numCols
        text(col, row, num2str(hadamardProduct(row, col),
'%d'), ...
            'HorizontalAlignment', 'center', ...
            'VerticalAlignment', 'middle', ...
            'FontSize',20);
    end
end
```



MATLAB code to calculate the Hadamard multiplication

Standard matrix multiplication

Standard Matrix Multiplication

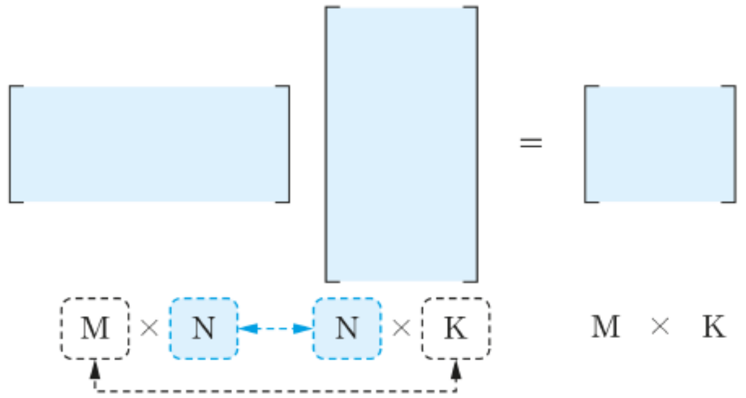
Characteristics of standard matrix multiplication

- Operating row/column-wise rather than operating element-wise.

- Reduces to a systematic collection of dot products.
  - Between rows of one matrix and columns of the other matrix.
  - Formally simply called $\boxed{\text{matrix multiplication.}}$
    - 'Standard' term is added to help disambiguate from Hadamard and scalar multiplications.

Rules for Matrix Multiplication Validity

- **First matrix sizes:** $M \times N$
- **Second matrix sizes:** $N \times K$
  - Multiplying these two matrices.
    - The " Inner " dimensions: N
    - The " Outer " dimensions: M and K
  - Matrix multiplication is valid only when the Inner dimensions match.
  - Size of product matrix is defined by outer dimensions.



Describing rules for matrix multiplication validity

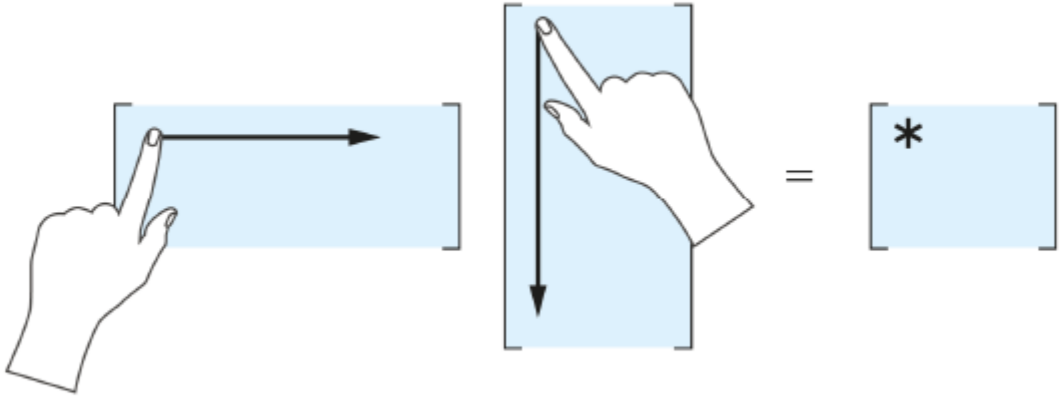
  - Matrix multiplication does not obey the commutative law.
    - If $\boldsymbol{C} = \boldsymbol{AB}$ and $\boldsymbol{D} = \boldsymbol{BA}$, then in general $\boldsymbol{C} \neq \boldsymbol{D}$.
    - They are equal in some special cases, but we cannot generally assume equality.

Matrix Multiplication

- **Why matrix multiplication is valid only if the number of columns in left matrix matches the number of rows in the right matrix?**
  - The $(i,j)th$ element in the product matrix is the dot product between the $i\ th$ row of the left matrix and the $j\ th$ column in the right matrix.
  - Dot product
    - A number that encodes the relationship between rows of the left matrix and columns of the right matrix.

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 2a + 3c & 2b + 3d \\ \boxed{\qquad\qquad} & 4b + 5d \end{bmatrix}$$

Example of multiplication



Finger movements for matrix multiplication

Result of Matrix Multiplication

- **Matrix that stores all the pairwise linear relationships between rows of the left matrix and columns for the right matrix.**

- **Can be a basis for computing covariance and correlation matrices.**
  - General linear model, singular-value decomposition, and countless other applications.

Matrices as linear transformations

Basic of Matrix-Vector Multiplication

- **A matrix can be right-multiplied by a column vector but not a row vector, and a matrix can be left-multiplied by a row vector but not a column vector.**
  - Matrix can be If $\boldsymbol{v}$ is column vector, $\boldsymbol{Av}$ and $\boldsymbol{v}^T\boldsymbol{A}$ are valid, but $\boldsymbol{Av}^T$ and $\boldsymbol{vA}$ are invalid.
  - $M \times N$ matrix can be pre-multiplied by a $1 \times M$ matrix or post-multiplied by an $N \times 1$ matrix.

- **Result of matrix-vector multiplication is always a vector.**
  - Orientation of that vector depends on the orientation of the multiplicand vector.
    - Pre-multiplying a matrix by a row vector produces another row vector.
    - Post-multiplying a matrix by a column vector produces another column vector.

Interpretations of Matrix-Vector Multiplication

Linear weighted combinations (more scalable method for computing)

- Put the individual vectors into corresponding elements of a vector, then multiply.

$$4\begin{bmatrix}3\\0\\6\end{bmatrix} + 3\begin{bmatrix}1\\2\\5\end{bmatrix} \Rightarrow \begin{bmatrix}3 & 1\\0 & 2\\6 & 5\end{bmatrix}\begin{bmatrix}4\\3\end{bmatrix}$$

Linear weighted combinations of column vectors

$$4\begin{bmatrix}3 & 0 & 6\end{bmatrix} + 3\begin{bmatrix}1 & 2 & 5\end{bmatrix} = \begin{bmatrix}4 & 3\end{bmatrix}\begin{bmatrix}3 & 0 & 6\\1 & 2 & 5\end{bmatrix}$$

Linear weighted combinations of row vectors

Geometric transforms

- Think of a vector as a geometric line.
  - Matrix-vector multiplication becomes a way of rotating and scailing that vector.
- Let's see the example of this on the next page.

Common Case in Matrix-Vector Multiplication

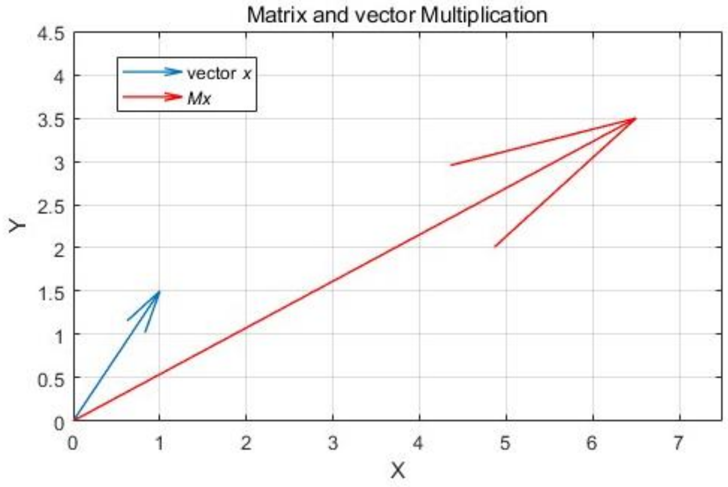
- **Set 2×2 matrix and 2×1 vector for multiplication**
  - Put the individual vectors into corresponding elements of a vector, then multiply.
  - Matrix $\boldsymbol{M}$ is set as $[2,3; 2,1]$, and the vector $\boldsymbol{x}$ is a vector set as $[1; 1.5]$.
  - Matrix $\boldsymbol{M}$ both rotated and stretched the original vector.
- **Main point of matrix-vector multiplication**
  - Matrix houses a transformation that can rotate and stretch that vector.

$$\boldsymbol{M} = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ 1.5 \end{bmatrix}$$

Matrix and vector

$$\boldsymbol{Mx} = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 6.5 \\ 3.5 \end{bmatrix}$$

Matrix and vector multiplication



Result of the multiplication

Uncommon Case in Matrix-Vector Multiplication

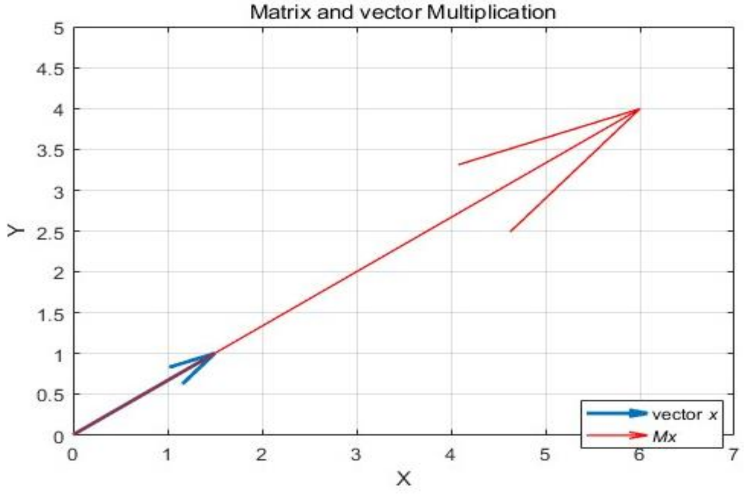
- **Set another vector as $x = [1.5; 1]$.**
  - Matrix-vector product is no longer rotated into a different direction.
  - Matrix-vector multiplication acted as if it were scalar-vector multiplication.
    - Note: Scalar-vector multiplication does not change the direction, but only changes magnitude
  - It's because vector x is an eigen vector of matrix M and the amount by which M stretched x is its eigen value.
  - This phenomenon will be explained later.

$$M = \begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} \qquad x = \begin{bmatrix} 1.5 \\ 1 \end{bmatrix}$$

Matrix and vector

$$Mx = \left(\begin{bmatrix} 2 \\ 2 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \right) \begin{bmatrix} 1.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

Matrix and vector multiplication

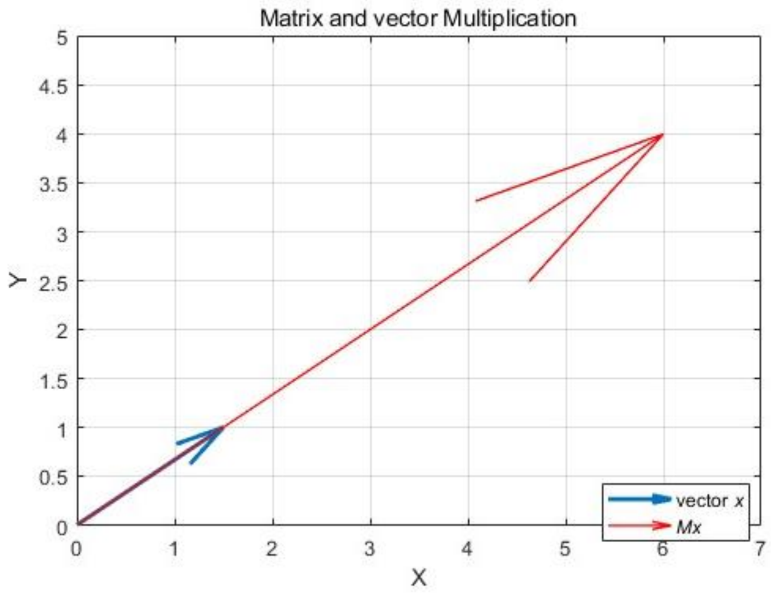


Result of the multiplication

Code Exercise of Matrix-Vector Multiplication

Code Exercise (05_09)

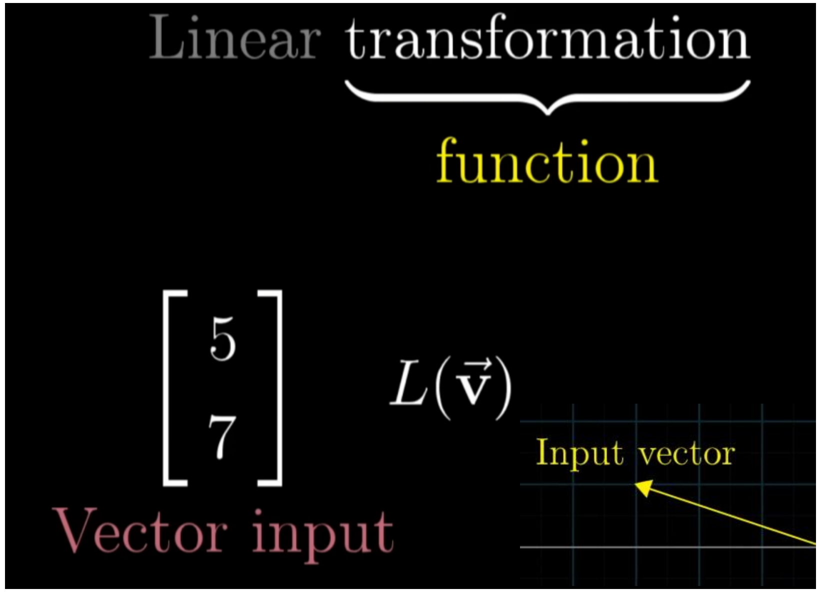
```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Create a 2x2 matrix with random values
matrix = [[2,3]; [2,1]];

% Create a 2x1 vector with random values
vector = [1.5; 1]; % Generate a 2x1 vector with integers between 1 and 10

% Calculate the matrix-vector multiplication
result = matrix * vector;

% Create and visualize the original 2x1 vector
figure;
quiver(0, 0, vector(1), vector(2), 'AutoScale', 'off', 'MaxHeadSize', 1,
'LineWidth', 2);
hold on; % Keep the same figure for the next quiver plot
axis equal; % Keep the x and y scales the same
grid on; % Add a grid for better readability
title('Matrix and vector Multiplication');
xlabel('X');
ylabel('Y');
xlim([0, max([vector(1), result(1)])+1]); % Set limits based on the larger
vector
ylim([0, max([vector(2), result(2)])+1]);

% Visualize the result of matrix-vector multiplication
quiver(0, 0, result(1), result(2), 'r', 'AutoScale', 'off', 'MaxHeadSize',
1, 'LineWidth', 1);

% Add a legend for clarity
legend('vector \itx', '\itMx', 'Location', 'Best');
hold off; % Release the figure for new plots
```



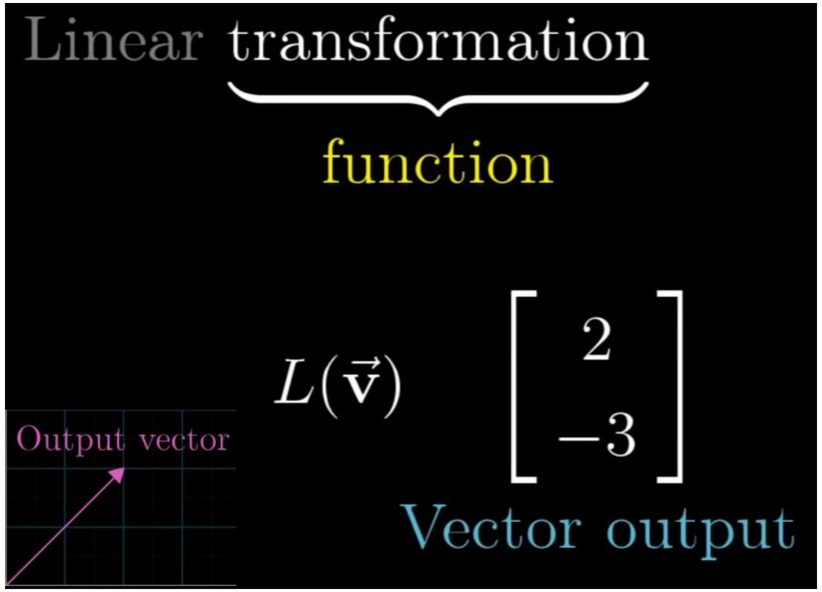
MATLAB code to calculate matrix-vector multiplication and result

What is Linear "Transformation"

- **The word transformation contains different meanings.**
  1. Function
     - It takes vector as an input, and outputs another vector.
  2. Movement
     - Input vector moves over the corresponding output vector.



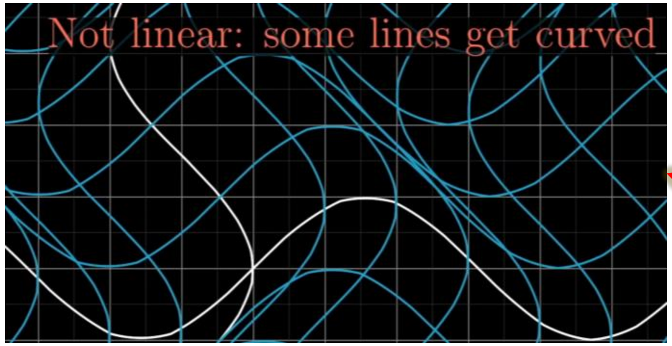
Input vector (w/visualization)



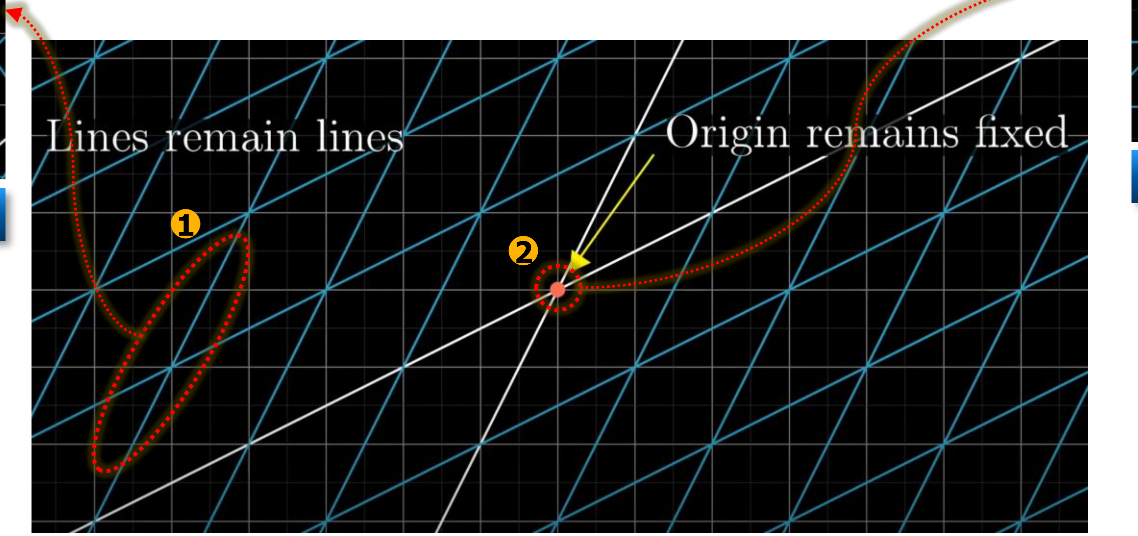
Output vector (w/visualization)

What is "Linear" Transformation

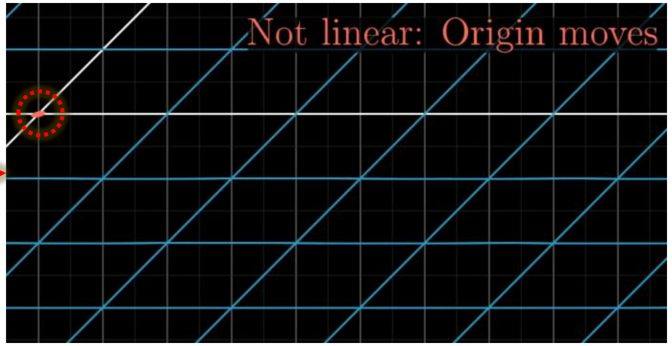
- **If a transformation has following two properties, it is linear.**
  1. All lines remain lines without getting curved.
  2. The origin must remain fixed in place.
- **Because of these properties, grid lines remain parallel and evenly spaced.**



Not linear : Lines get curved



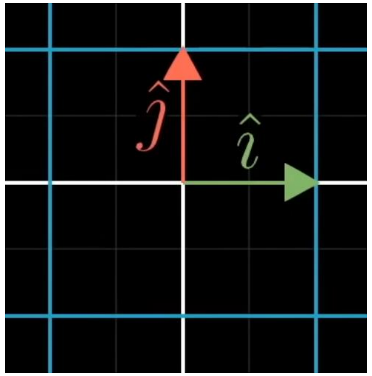
Example of Linear Transformation



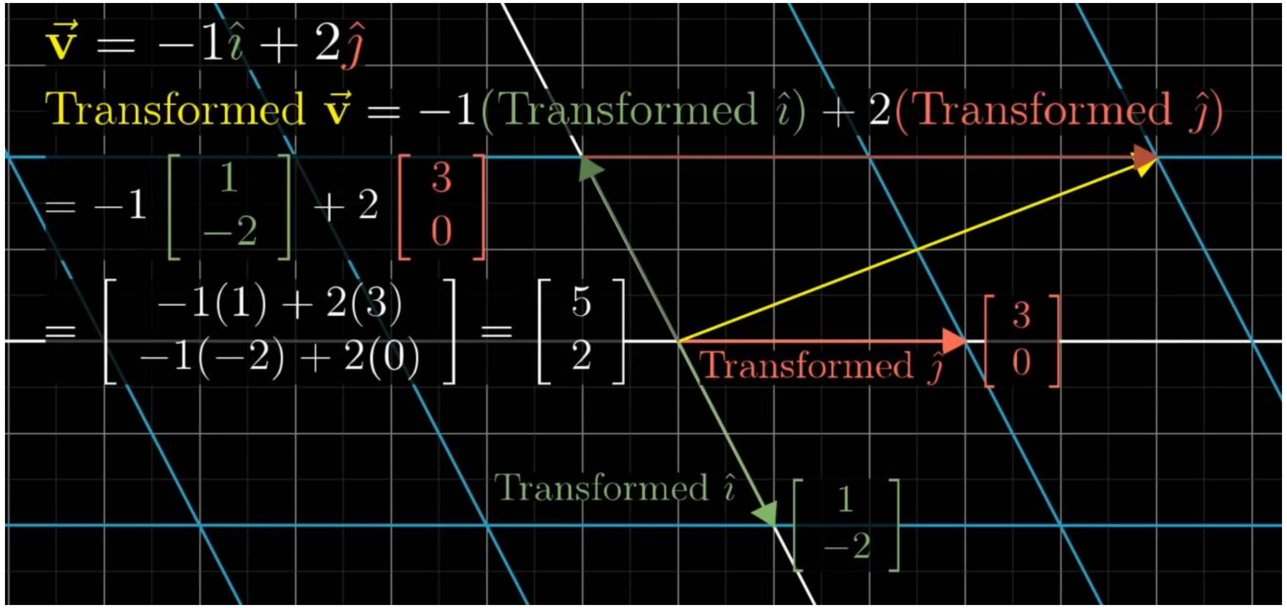
Not linear : Origin moves

Numerical Example of Linear Transformation

- **Only consider where the two basis vectors $\hat{\imath}$, and $\hat{\jmath}$ each land.**
- **Example**
  - Set vector as $\mathbf{v} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$, meaning that $\mathbf{v}$ equals $-1 \times \hat{\imath} + 2 \times \hat{\jmath}$.
  - After transformation, there is a property that grid lines remain parallel and evenly spaced.
    - $\mathbf{v}$ lands will be $-1 \times$ the vector where $\hat{\imath}$ landed $+2 \times$ the vector where $\hat{\jmath}$ landed.
  - So, if started off as a certain linear combination of $\hat{\imath}$ and $\hat{\jmath}$, then ends up as that same $\boxed{\text{linear Combination}}$ of where those two vectors landed.



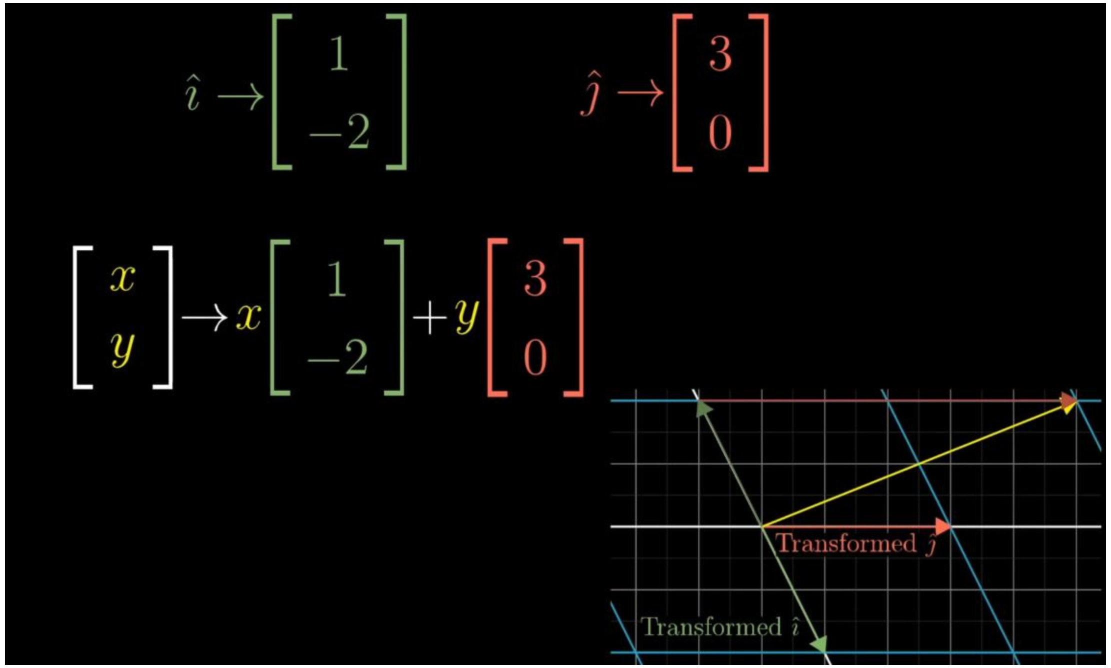
Basis vector, $\hat{\imath}$ and $\hat{\jmath}$



Visualization of Example (Linear Transform)

Generalization of Linear Transformation

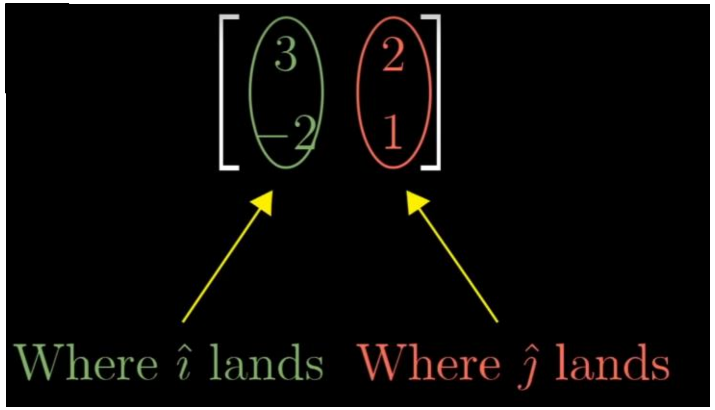
- **Write the example with more general coordinate, x and y.**
  - x will land on x times the vector where $\hat{\imath}$ lands, $\begin{bmatrix} 1 \\ -2 \end{bmatrix}$, plus y times the vector where $\hat{\jmath}$ lands, $\begin{bmatrix} 3 \\ 0 \end{bmatrix}$.
  - The summation will land at $\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow x \begin{bmatrix} 1 \\ -2 \end{bmatrix} + y \begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1x + 3y \\ -2x + 0y \end{bmatrix}$.
- **So, given any vector, we can calculate where that vector lands using this formula.**



General Formula of Linear Transformation

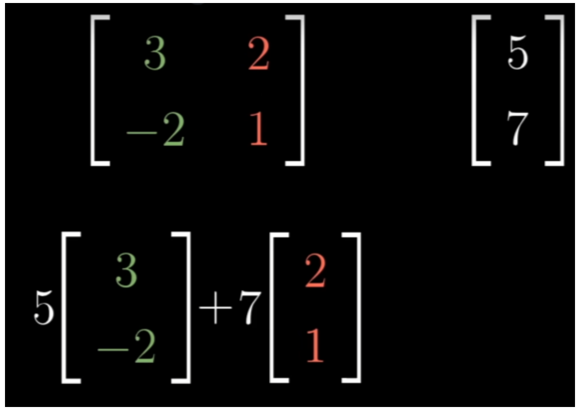
Numerical Example of Linear Transformation in 2×2 Matrix

- **It is common to package coordinate into a 2×2 matrix**
  - Column 0 is where $\hat{\imath}$ lands
  - Column 1 is where $\hat{\jmath}$ lands
- **If we expand linear transformation into 2×2 matrix form,**
  - Take the coordinates of the vector, multiply them by the corresponding columns of the matrix, then add together.

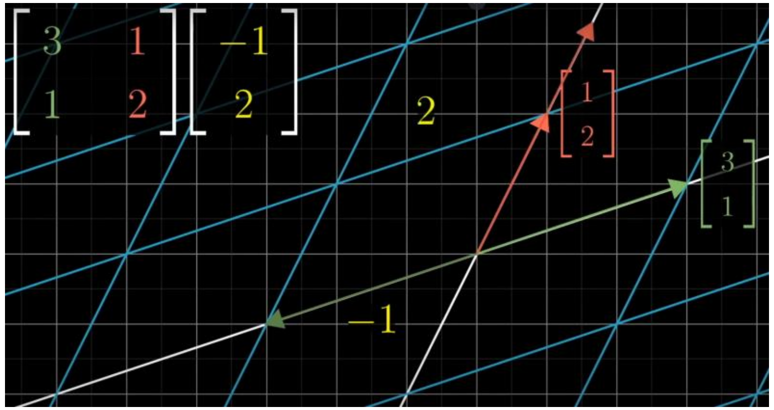


2x2 Matrix Representation of Coordinates

Numeric



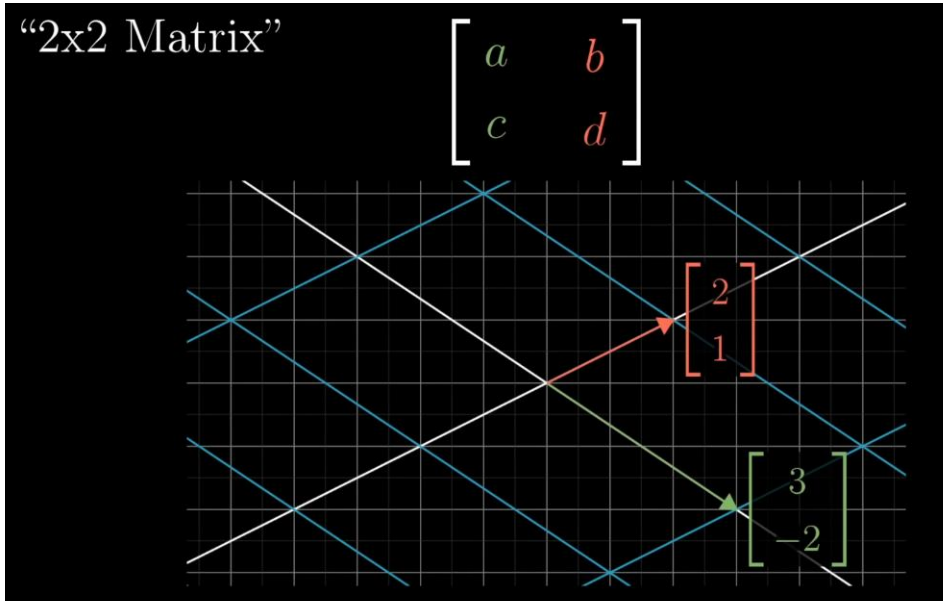
Visual



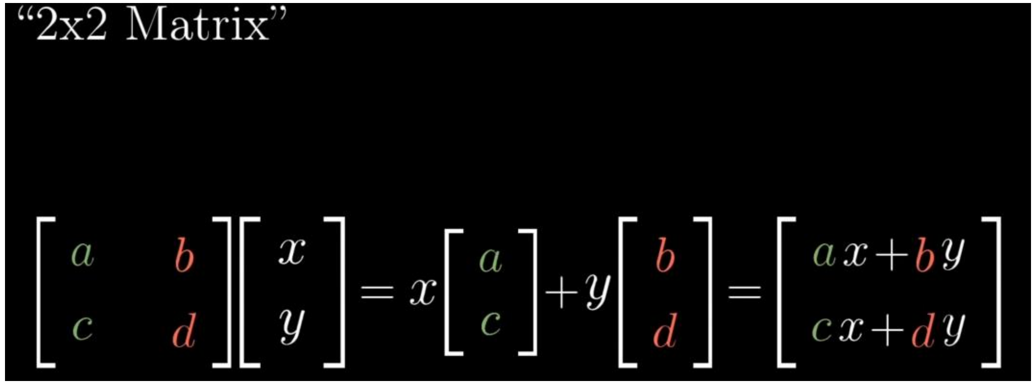
Calculation of Linear Transformation in 2x2 Matrix

Generalization of Linear Transformation in 2×2 Matrix

- **Write the example in general case, where your matrix has entries $\boldsymbol{a, b, c, d}$.**
  - First column, $\boldsymbol{a, c}$, as the place where the first basis vector lands.
  - Second column, $\boldsymbol{b, d}$, as the place where the second basis vector lands.



2x2 Matrix Representation of Coordinates

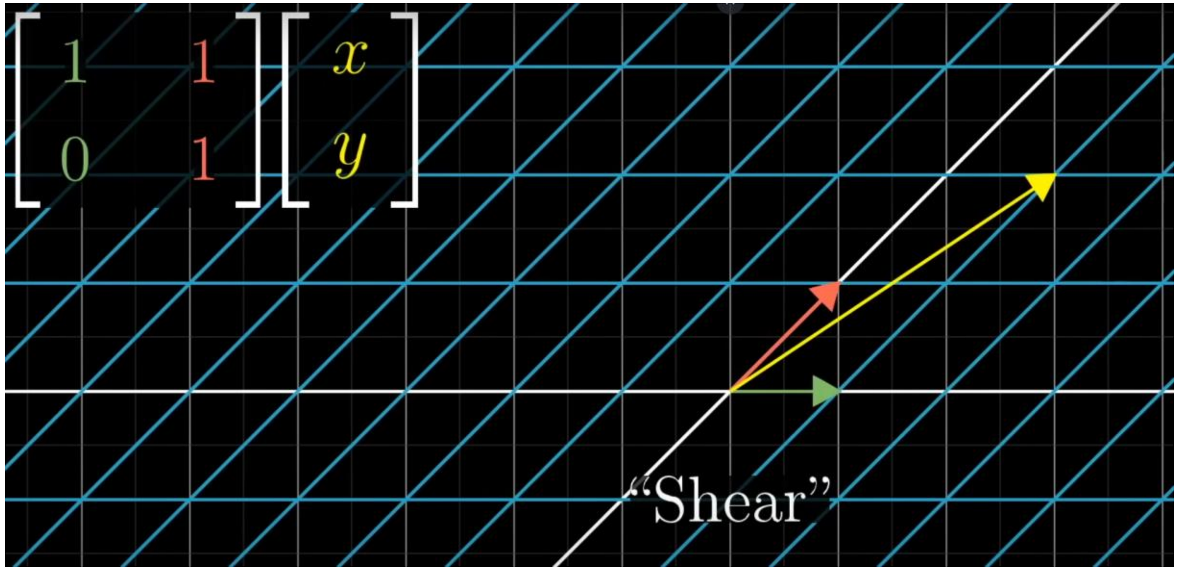


2x2 Matrix Representation of Coordinates

Example of Transformation

Shear transformation

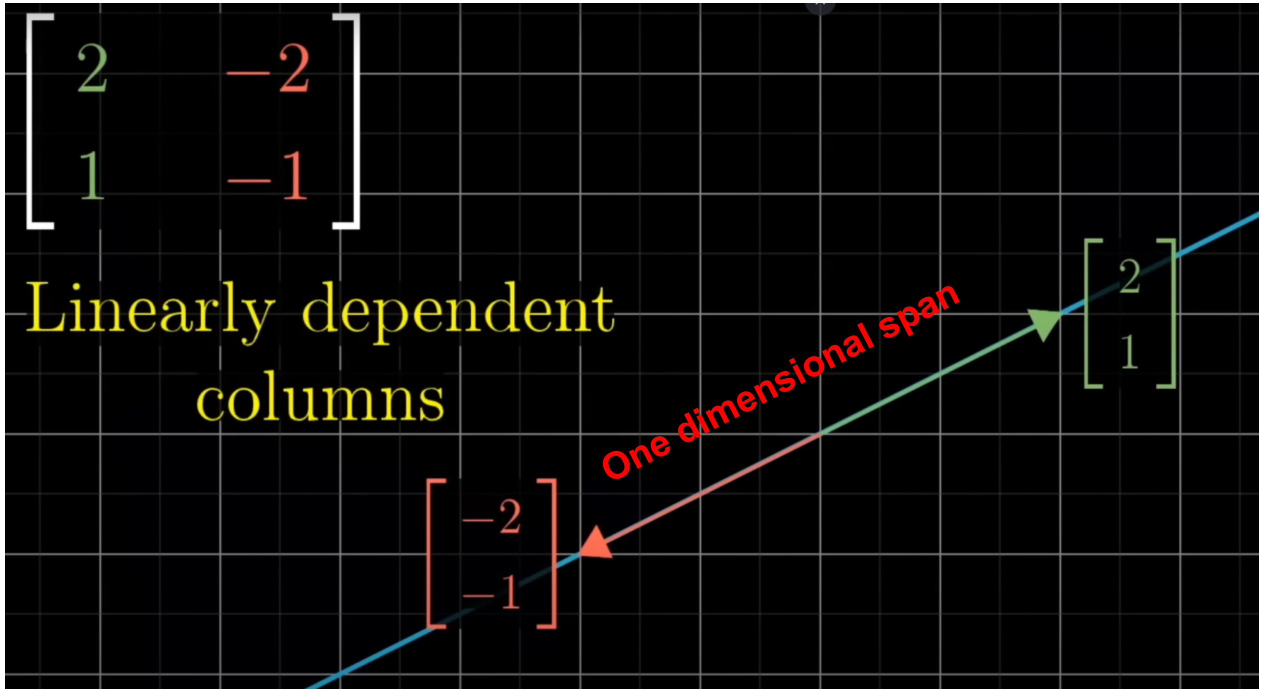
- $\hat{\imath}$ remains fixed, so the first column of the matrix is $\begin{bmatrix}1\\0\end{bmatrix}$.
- $\hat{\jmath}$ moves over to the coordinates $\begin{bmatrix}1\\1\end{bmatrix}$, which become the second column of the matrix.
- Knows how a shear transforms a vector by multiplying this matrix $\begin{bmatrix}1 & 1\\0 & 1\end{bmatrix}$ by vector $\begin{bmatrix}x\\y\end{bmatrix}$.



Shear transformation

Linear Transformation in Linearly Dependent Columns

- **If the vectors that i-hat and j-hat land on are** linearly dependent, **one vector is scaled version of another vector.**
  - Linear transformation squishes all of 2D space onto the line, where those two vectors sit.
  - This 2D space is called the one-dimensional span of those two linearly dependent vectors.



Span of two linearly dependent columns

Matrix operation: transpose

Transpose Operation on Matrix

Principle of Transpose Operation

- Simply swap the $\boxed{\text{rows and columns}}$

Notation of Transpose Operation

- Indicate with a superscripted ^{T}.
- Double-transposing a matrix returns the original matrix. ($\boldsymbol{C}^{TT} = \boldsymbol{C}$)

$$\boldsymbol{a}_{i,j}^{T} = \boldsymbol{a}_{j,i}$$

Definition of the transpose operation

$$\begin{bmatrix} 3 & 0 & 4 \\ 9 & 8 & 3 \end{bmatrix}^{T} = \begin{bmatrix} 3 & 9 \\ 0 & 8 \\ 4 & 3 \end{bmatrix}$$

Example of the transpose operation

Code Exercise of Transpose Operation on Matrix

A few ways to transpose matrices in MATLAB

- Function < ' >
- Function <transpose(A)>

Code Exercise (05_10)

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Create a 3x2 matrix with random values
matrix = randi(10, 3, 2); % Generate a 3x2 matrix with integers between 1
and 10

% Transpose the matrix
transposedMatrix = matrix';
transposedMatrix2 = transpose(matrix);

% Create and visualize the original matrix as vectors
disp("Original Matrix");
disp(matrix);
disp("Transposed Matrix using (')");
disp(transposedMatrix);
disp("Transposed Matrix using (transpose(A))");
disp(transposedMatrix2);
```

```
Original Matrix
     8     1
     4     7
     3     5

Transposed Matrix using (')
     8     4     3
     1     7     5

Transposed Matrix using (transpose(A))
     8     4     3
     1     7     5
```

MATLAB code to transpose operation on matrix

Dot and Cross Product Notation

Dot product of vectors

- Vector $\boldsymbol{a}: 2 \times 1$, vector $\boldsymbol{b}: 2 \times 1$.
- The dot product is indicated as $\boxed{\boldsymbol{a}T\boldsymbol{b}}$
  - The "inner" dimensions match, and the "outer" dimensions will be $\mathbf{1} \times \mathbf{1}$ as Eq 1..

Cross product of vectors

- Vector $\boldsymbol{a}: 2 \times 1$, vector $\boldsymbol{b}: 3 \times 1$.
- The cross product is indicated as $\boldsymbol{a}\boldsymbol{b}^{T}$.
  - The "inner" dimensions match, and the "outer" dimensions will be $\mathbf{2} \times \mathbf{3}$ as Eq 2..

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \qquad \boldsymbol{a}^T\boldsymbol{b} = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = [a_1 b_1 + a_2 b_2]$$

Eq 1. Example of dot product of vectors

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \qquad \boldsymbol{a}\boldsymbol{b}^T = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \end{bmatrix}$$

Eq 2. Example of cross product of vectors

Matrix operation: LIVE EVIL (order of operation)

The Transpose of Multiplied Matrices

- **LIVE EVIL is a palindrome.**
  - Palindrome is a word or phrase that is spelled the same forwards and backwards.

- **Transpose of multiplied matrices is same as the individual matrices transposed and multiplied.**
  - But reversed in order as shown in below equation.

- **Assume**
  - $\boldsymbol{L}, \boldsymbol{I}, \boldsymbol{V}$ and $\boldsymbol{E}$ are all matrices.
  - Their sizes match to make multiplication valid.

$$(\boldsymbol{LIVE})^T = \boldsymbol{E}^T \boldsymbol{V}^T \boldsymbol{I}^T \boldsymbol{L}^T$$

Example of the LIVE EVIL rule

- **This rule applies for multiplying any number of matrices.**

Symmetric matrix

Definition of Symmetric Matrix

- **The corresponding rows and columns are equal.**
  - When you $\boxed{\text{swap}}$ the rows and columns, nothing happens to the matrix!
- **A symmetric matrix equals its transpose, $A^T = A$.**
- **Then, can non-square matrix be symmetric?**
  - Nope! Why can't be?
    - If matrix is of size M*N, then its transpose is of size N*M.
    - It cannot be guaranteed that M and N are always the same value.

$$\boldsymbol{A} = \begin{bmatrix} a & e & f & g \\ e & b & h & i \\ f & h & c & j \\ g & i & j & d \end{bmatrix}$$

A symmetric matrix

Creating Symmetric Matrix from Nonsymmetric Matrix

- **Multiplying any matrix by its transpose, it will be a square symmetric matrix as eq 1..**

- **Prove symmetry**
  - Recall that the definition of a symmetric matrix is one that equals its transpose as eq 2..
  - The proof relies on the LIVE EVIL rule.

- **But AA^T and A^TA are square symmetric, but** not same **matrix.**
  - If $\boldsymbol{A}$ in non-square, then two matrix products are not even the same size.

$$if\ \boldsymbol{A} = M \times N, \qquad \boldsymbol{A}^T\boldsymbol{A} = (N \times M)(M \times N) = N \times N$$

Eq 1. Symmetric matrix from nonsymmetric matrix

$$(\boldsymbol{A}^T\boldsymbol{A})^T = \boldsymbol{A}^T\boldsymbol{A}^{TT} = \boldsymbol{A}^T\boldsymbol{A}$$

Eq 2. Definition of a symmetric matrix

Summary

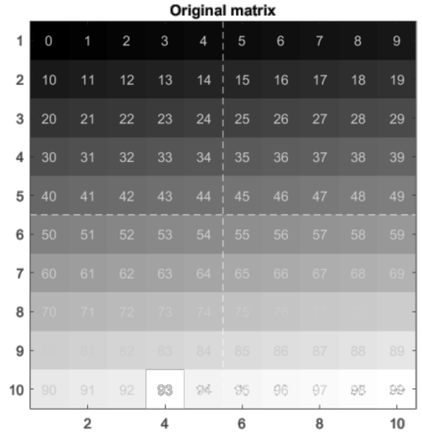
Summary

- **Matrix**
  - Spreadsheet of numbers.
- **Several categories of special matrices**
  - Random number, square, non-square, diagonal, triangular, identity and zeros matrix.
- **Some arithmetic operations that work element-wise**
  - Addition, scalar multiplication and Hadamard multiplication.
- **Shifting a matrix**
  - Adding a constant to the diagonal elements.
- **Matrix multiplication validity**
  - First matrix sizes are $M \times N$, second matrix sizes are $N \times K$.
- **The transpose of multiplied matrices**
  - The individual matrices transposed and multiplied with their order reversed.
- **Symmetric matrix**
  - Each row equals its corresponding columns, $\boldsymbol{A} = \boldsymbol{A}^T$.
  - Create from any matrix by multiplying that matrix by its transpose.

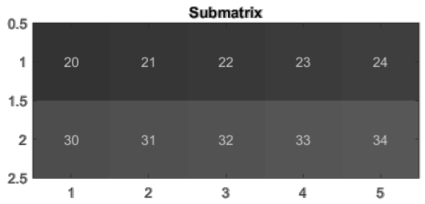
Exercise: Matrix Slicing

- **Create original matrix**
  - Shape: $10\ x\ 10$
  - Value: 0-99 (Refer to the figure(Result of code))
- **Create function**
  - Input: Original matrix, start row index, end row index, start col index, end col index
  - Output: Submatrix from Original matrix
    - Output can be express various type.

Input



Output



Result of code

Exercise: Matrix Addition

- **Use for loops for rows and columns to implement matrix additions for each element.**
  - Input: Two matrix with same size
  - Output: The sum of two matrix
  - Use the error() function in the MATLAB to make the error generate if the size of the two input matrix is different.

Exercise: Matrix Multiplication

- **Use for loops for rows and columns to implement dot product.**
  - Input: Two matrix with same size
  - Output: The sum of two matrix
  - Use the error() function in the MATLAB to make the error generate if dot product is not available.

Exercise: Symmetric Checker

- **Create a function that checks whether the input matrix is symmetrical or not.**
  - Input: a matrix
  - Output: 1 (if symmetric) / 0 (if not symmetric)

THANK YOU
FOR YOUR ATTENTION