

COMPLETE LINEAR ALGEBRA: THEORY AND IMPLIMENTATION

Course: <https://www.udemy.com/course/linear-algebra-theory-and-implementation/>

Instructor: Mike X Cohen: <http://sincxpress.com/>

Notes and code completion (student): Ron Fredericks: <https://BiophysicsLab.com/>

Start: May 19, 2020 End: TBD

CODING LANGUAGES USED:

MATLAB (and/or Python with Jupiter)

MATLAB TOOLBOXES USED:

Lesson 56 Multiplication of two symmetric matrices: Symbolic Toolbox

PYTHON LIBRARIES USED:

numpy

sympy

CONTENTS

COMPLETE LINEAR ALGEBRA: THEORY AND IMPLIMENTATION	1
Coding languages used:	1
MATLAB Toolboxes used:	1
Python Libraries used:	1
Contents	1
Figures	5
SECTION 1: COURSE INTRODUCTION	8
1. What Is Linear Algebra	8
2. Linear algebra applications	9
3. An enticing start to a linear Algebra Course	10
Section 3: Vectors	12
8. Algebraic and geometric interpretations of vectors	12
9. Vector addition and subtraction	14
10. Vector-scalar multiplication	17
11. Vector - Vector multiplication: the dot product	18
12. Dot product properties: associative, distributive, commutative	19
13. Code challenge: dot products with matrix columns	20
14. Vector length	21
15. Dot product geometry: sign and orthogonality	21
16. Code challenge: dot product sign and scalar multiplication	23
17. Code challenge: Cauchy-Schwarz inequality	24
18. Code challenge: is the dot product commutative?	25

19. Vector Hadamard multiplication	26
20. Outer Product	26
21. Vector cross product.....	27
22. Vectors with complex numbers	29
23. Hermitian transpose (a.k.a. conjugate transpose)	30
24. Interpreting and creating unit vectors.....	32
25. Code Challenge: Dot Products with Unit Vectors	33
26. Dimensions and fields in linear algebra	33
27. Subspaces	34
28. Subspaces vs. subsets	35
29. Span	35
Quiz 5: In the span?	36
30. Linear independence	37
31. Basis	39
Section 4: Introduction to matrices	40
33: Matrix terminology and dimensionality	40
34. A zoo of matrices	40
35. Matrix addition and subtraction	41
36. Matrix-scalar multiplication.....	43
37. Code challenge: is matrix-scalar multiplication a linear operation?.....	43
38. Transpose.....	44
39. Complex matrices	44
40. Diagonal and trace.....	45
41. Code challenge: linearity of trace	47
42. Broadcasting matrix arithmetic	47
Section 5: Matrix multiplications.....	48
44. Introduction to standard matrix multiplication	48
45. Four ways to think about matrix multiplication	50
46. Code challenge: matrix multiplication by layering	52
47. Matrix multiplication with a diagonal matrix	53
48. Order-of-operations on matrices.....	53
49. Matrix-vector multiplication.....	54
50. 2D transformation matrices.....	57
51. Code challenge: Pure and impure rotation matrices	60

52. Code challenge: Geometric transformations via matrix multiplications	61
53. Additive and multiplicative matrix identities.....	63
54. Additive and multiplicative symmetric matrices	64
55. Hadamard (element-wise) multiplication.....	65
56. Code challenge: symmetry of combined symmetric matrices.....	66
57. Multiplication of two symmetric matrices.....	67
58. Code challenge: standard and Hadamard multiplication for diagonal matrices	68
59. Code challenge: Fourier transform via matrix multiplication	69
60. Frobenius dot product	71
61. Matrix norms	72
62. Code challenge: conditions for self-adjoint	73
63. Code challenge: The matrix asymmetry index.....	74
64. What about matrix division?	76
Section 6: Matrix rank.....	77
66. Rank: concepts, terms, and applications	77
67. Computing rank: theory and practice	79
68. Rank of added and multiplied matrices.....	80
69. Code challenge: reduced-rank matrix via multiplication	80
70. Code challenge: scalar multiplication and rank	81
71. Rank of $A^T A$ and $A A^T$	81
72. Code challenge: rank of multiplied and summed matrices	82
73. Making a matrix full-rank by “shifting”	83
74. Code challenge: is this vector in the span of the set	85
Section 7: Matrix spaces	86
77. Column space of a matrix	86
78. Column space, visualized in code	87
79. Row space of a matrix.....	88
80. Null space and left null space of a matrix	89
81. Column/left-null and row/null spaces are orthogonal	91
82. Dimensions of column/row/null spaces	91
83. Example of the four subspaces.....	91
84. More on $Ax=b$ and $Ax=0$	93
Section: 8: Solving systems of equations.....	94
86. Systems of equations: Algebra and geometry	94

87. Converting systems of equations to matrix equations	97
88. Gaussian elimination	98
89. Echelon form and pivots	101
90. Reduced row echelon form	102
91. Code challenge: RREF of matrices with different sizes and ranks	103
92. Matrix spaces after row deduction.....	104
Section 9: Matrix determinant	105
94. Determinant: concept and applications.	105
95. Determinant of a 2x2 matrix.....	106
96. Code challenge: determinant of small and large singular matrices.....	106
97. Determinant of a 3x3 matrix.....	107
98. Code challenge: large matrices with row exchanges.....	108
99. Find matrix values for a given determinant.....	109
100. Code challenge: determinant of shifted matrices	109
101. Code challenge: determinant of matrix product	111
Section 10: Matrix inverse	112
103. Matrix inverse: Concept and applications	112
104. Computing the inverse in code.....	113
105. Computing the 2x2 matrix inverse.....	114
106. The MCA algorithm to compute the matrix inverse	115
107. Code challenge: Implement the MCA algorithm	118
108. Computing the inverse via row reduction	120
109. Code challenge: inverse of a diagonal matrix.....	123
110. Left inverse and right inverse	125
111. One-sided inverses in code.....	125
112. Proof: the inverse is unique.....	127
113. Pseudo-inverse, part 1	128
115. Why should you avoid the inverse?.....	131
Section 11: Projections and orthogonalization.....	131
108. Projections in \mathbb{R}^2	131
109. Projections in \mathbb{R}^N	134
110. Orthogonal and Parallel Vector Components.....	137
111. Code challenge Decompose vector	139
112. Orthogonal matrices.....	141

Glossary	141
References	142

FIGURES

Figure 1 - Use vectors to represent sensor data.....	9
Figure 2 – MATLAB singular value components and spectrum demonstration	11
Figure 3 - Two-dimensional vector	12
Figure 4 - Vector valued functions.....	13
Figure 5 – MATLAB vector plot in 2D and 3D demonstration.....	14
Figure 6 - Vector addition: Geometry	15
Figure 7 - Vector subtraction: Geometry (2 methods)	16
Figure 8 - MATLAB vector addition demonstration.....	17
Figure 9 – MATLAB vector-scalar multiplication demonstration.....	18
Figure 10 - Dot product example	19
Figure 11 - Vector length via geometry (and via Pythagoras)	21
Figure 12 - How to compute the angle between any two vectors	22
Figure 13 - Vector algebra vs. geometry.....	22
Figure 14 - MATLAB dot product demonstration	23
Figure 15 - The Cauchy-Schwarz inequality	24
Figure 16 - Hadamard multiplication.....	26
Figure 17 - Outer product.....	27
Figure 18 - Cross-product: algebraic formula	28
Figure 19 Cross-product: Geometric formula.....	28
Figure 20 - MATLAB 3D plot of vector cross product	29
Figure 21 - Vector with real and imaginary dimensions.....	30
Figure 22 - Dot product with complex vectors example.....	30
Figure 23 - Hermitian transpose	31
Figure 24 - Hermitian transpose example: Length of vector	31
Figure 25 - MATLAB unit vector example	33
Figure 26 - Ambient spaces and subspaces: Geometry and Algebra.....	34
Figure 27 - Span: algebraic example	35
Figure 28 - MATLAB span demonstration.....	36
Figure 29 Quiz 5: In the span?	37

Figure 30 - Example of linearly dependent sets	38
Figure 31 - Linear independence: geometry.....	38
Figure 32 - Example of standard basis vectors	39
Figure 33 - "Shifting" a matrix away from degeneracy.....	42
Figure 34 - Complex matrices	45
Figure 35 - Diagonal.....	45
Figure 36 – Trace	46
Figure 37 - Standard matrix multiplication: The rules for validity	49
Figure 38 - Matrix multiplication - element perspective	50
Figure 39 - Matrix multiplication - layer perspective.....	51
Figure 40 - Matrix multiplication - column perspective	51
Figure 41 - Matrix multiplication - row perspective	52
Figure 42 - Four ways to think about matrix multiplication	52
Figure 43 - Matrix multiplication with a diagonal matrix	53
Figure 44 - Matrix-vector multiplication: always a vector	55
Figure 45 - Matrix-vector multiplication example	56
Figure 46 - Matrix-vector multiplication with symmetry.....	56
Figure 47 - 2D transformation matrices: rotation and stretch	57
Figure 48 - 2D transformation matrices: rotation	58
Figure 49 - 2D transformation matrices: stretch	58
Figure 50 - MATLAB 2D rotation and stretch transformation matrix $Av: v = [3 \ -2]; A = [1 \ -1; 2 \ 1]$	59
Figure 51 - MATLAB 2D rotation transformation matrix $Av: v = [3 \ -2]; A = [\cos(\theta) \ -\sin(\theta); \sin(\theta) \ \cos(\theta)]$	59
Figure 52 - MATLAB Code challenge: Pure and impure rotation matrices	61
Figure 53 - MATLAB Code challenge: Geometric transformations via matrix multiplications	63
Figure 54 - Symmetric matrix via the additive method: example	64
Figure 55 - Multiplicative symmetric matrix method: example	65
Figure 56 - Hadamard (element-wise) multiplication.....	66
Figure 57 - Symmetric matrices produce asymmetric result.....	67
Figure 58 - MATLAB Fourier Transform results	70
Figure 59 - MATLAB Sine (imaginary) and Cosine (real) results.....	70
Figure 60 - MATLAB view real components of matrix using <code>imagesc</code>	71
Figure 61 - Test on random matrices.....	76
Figure 62 - Hadamard (element-wise) division.....	77
Figure 63 - Rank terminology.....	78

Figure 64 - Rank dimensionality of information: algebraic and geometric	78
Figure 65 - Difficulties in computing rank of matrices in practice	79
Figure 66 - "Shifting" a matrix: normal example	84
Figure 67 - View a symmetric matrix with imagesc	85
Figure 68 – Verify vector is in column space of matrix	86
Figure 69 - Determine how close vector is to column space of matrix	87
Figure 70 – MATLAB demonstration of a vector v (red) contained in S (green plane)	88
Figure 71 - Column space vs. Row space	89
Figure 72 - Null space of a matrix: relation to independence	90
Figure 73 - Null space of a matrix: Graphic interpretation	90
Figure 74 - Picture of the four subspaces	92
Figure 75 - Example of span as one of the four subspaces	93
Figure 76 - Example of the four subspaces	93
Figure 77 - $Ax=b$: Is there an exact solution?	94
Figure 78 - Systems of equations: algebra and geometry	95
Figure 79 - MATLAB 2d linear system of equations demonstration	96
Figure 80 - MATLAB 3d linear system of equations demonstration	97
Figure 81 - Converting systems of equations to matrix equations	98
Figure 82 - Convert system of equations to matrices	98
Figure 83 - Augment coefficients matrix with constants	98
Figure 84 - Convert matrix into an upper-triangular matrix	99
Figure 85 - Map matrix back onto equations	99
Figure 86 - Back-substitution to solve for variables	99
Figure 87 - Echelon form and pivots	101
Figure 88 - Echelon form conversion: example	102
Figure 89 - Reduced row echelon form: examples	102
Figure 90 - Reduced row echelon form: worked problem	103
Figure 91 - MATLAB Matrix spaces after row reduction	105
Figure 92 - Determinant of a 3x3 matrix: visual trick #1	107
Figure 93 - Determinant of a 3x3 matrix: visual trick #2	108
Figure 94 - MATLAB code challenge: determinant of shifted matrices demonstration	111
Figure 95 - demonstrate numerical instabilities when computing determinant products	112
Figure 96 - Use imagesc to demonstrate matrix times its inverse is the identity matrix	114
Figure 97 - Four-step plan to compute matrix inverse	115

Figure 98 - The minors matrix: a matrix of determinants.....	116
Figure 99 - The MCA algorithm: The cofactors matrix +/- checkerboard.....	116
Figure 100 - The MCA algorithm: The cofactors matrix.....	117
Figure 101 - The MCA algorithm: The adjugate matrix	117
Figure 102 - MCA algorithm example: matrix inverse	118
Figure 103 - imagesc of resulting identity matrix	120
Figure 104 - Computing inverse via row reduction	121
Figure 105 Computing inverse via row reduction: example 1	121

SECTION 1: COURSE INTRODUCTION

See included exercises with code download:

- linalg_picSVD.m
- linalg_picSVD.ipynb

1. WHAT IS LINEAR ALGEBRA

Example where used:

- Geometry
- Calculus
- Differential equations
- Computing
- Statistics
- Computer graphics
- Modern scientific numerical statistical applications

Vector: an ordered list of numbers or functions:

$$\begin{pmatrix} 1 \\ 0 \\ 2 \\ 5i \\ -2 \end{pmatrix} \quad (1 \ 2 \ 3) \quad \begin{pmatrix} \sin(t) \\ \cos(t) \\ t \end{pmatrix}$$

Matrix: a “spreadsheet” of numbers:

$$\begin{pmatrix} 2 & 0 & 4 & 10 & -4 \\ 8 & 0 & 16 & 40 & -16 \\ -6 & 0 & -12 & -30 & 12 \\ 1 & 0 & 2 & 5 & -12 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Vectors and matrices have both algebraic and geometric interpretations.

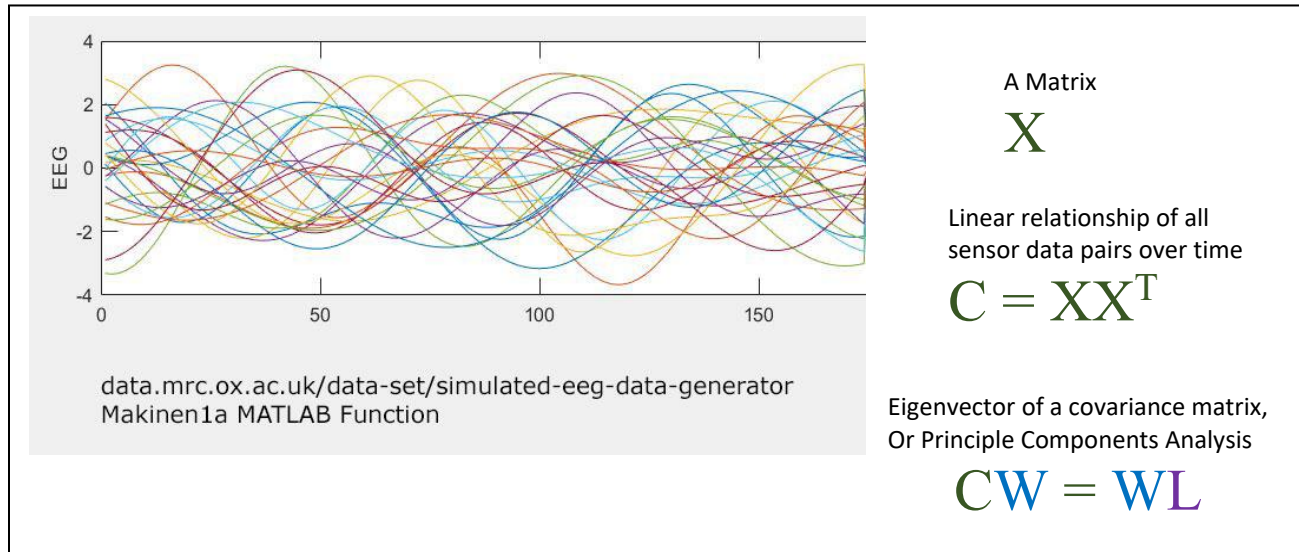


Figure 1 - Use vectors to represent sensor data

Normalized Quadratic Form

“Linear algebra”: focus on theory and proofs

“Matrix algebra”: focus on applications

2. LINEAR ALGEBRA APPLICATIONS

Convert data from rows and columns into a matrix X

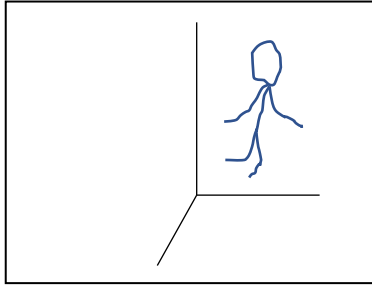
$$\begin{aligned}
 s_1 &= \beta_1 c_1 + \beta_2 \\
 s_2 &= \beta_1 c_2 + \beta_2 \\
 s_3 &= \beta_1 c_3 + \beta_2 \\
 s_4 &= \beta_1 c_4 + \beta_2 \\
 s_5 &= \beta_1 c_5 + \beta_2
 \end{aligned}
 \Rightarrow
 \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{bmatrix}
 =
 \begin{bmatrix} c_1 & 1 \\ c_2 & 1 \\ c_3 & 1 \\ c_4 & 1 \\ c_5 & 1 \end{bmatrix}
 \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}
 \Rightarrow y = X\beta$$

$$\beta = (X^T X)^{-1} X^T y$$

Manipulate pixels forming an image on a 3D screen

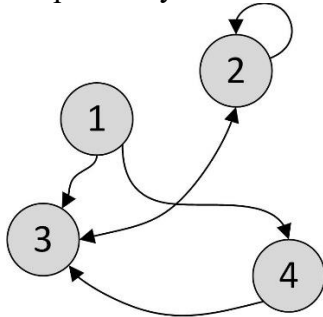
Rotation matrix

Alternative method
Quaternion



$$\begin{matrix} \cos(\theta) & -\sin(\theta) & 0 & a \\ \sin(\theta) & \cos(\theta) & 0 & bi \\ 0 & 0 & 1 & cj \\ & & & dk \end{matrix}$$

Graph theory



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

3. AN ENTICING START TO A LINEAR ALGEBRA COURSE

Example of singular value decomposition

MATLAB

Code: linalg_picSVD.m

Image: Einstein_tongue.jpg

SINGULAR VALUE DECOMPOSITION

Load image of Einstein's Tongue

Take singular value decomposition (MATLAB function SVD) of 2D image matrix using gray colormap

Plot singular values

Compare image regenerated from a range of singular value components ranked from most significant to least significant

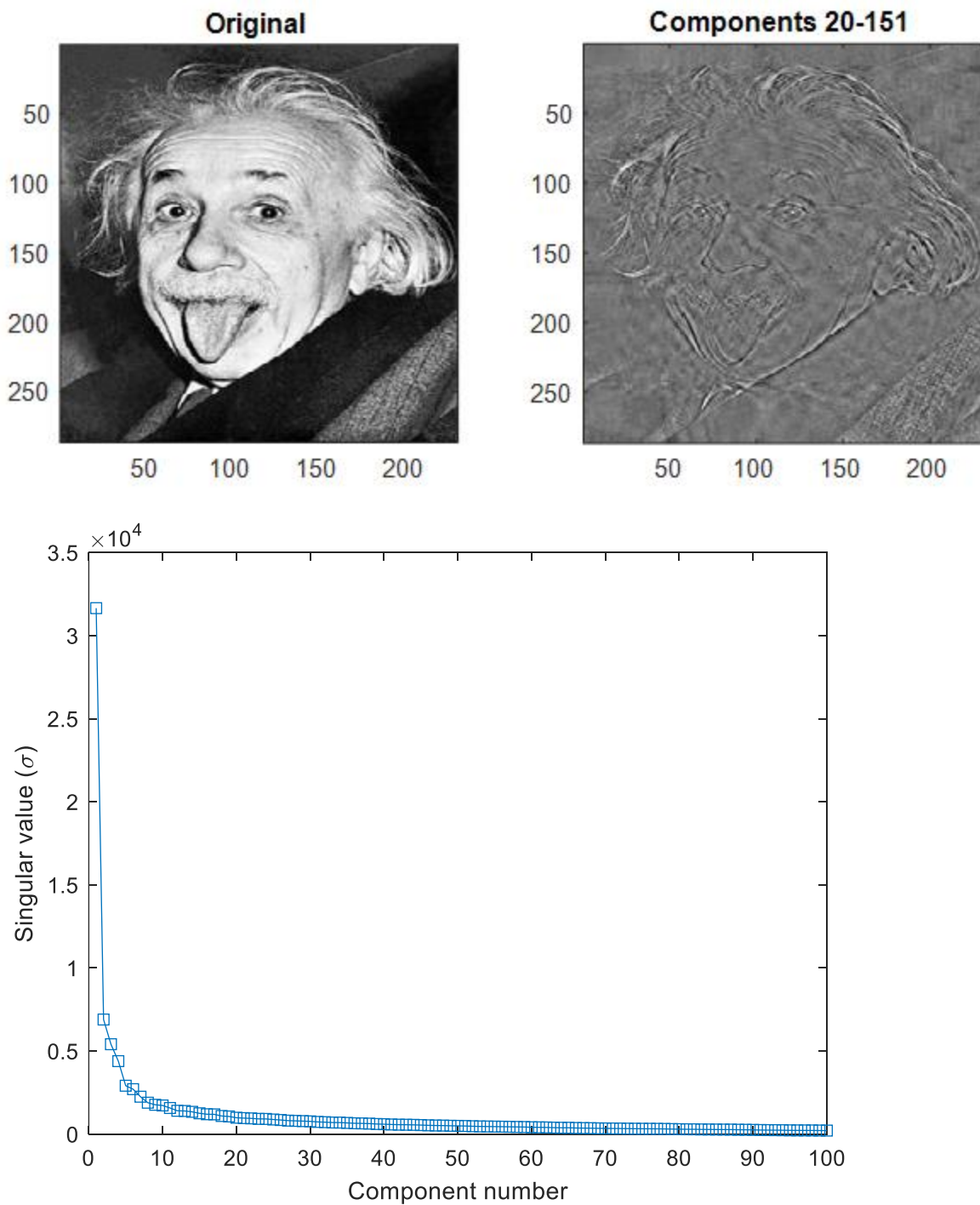


Figure 2 – MATLAB singular value components and spectrum demonstration

Deconstruct a matrix into three matrices showing important patterns (special, direction, patterns of covariance). Where the first 4 components are large compared to the others. These first few components can reconstruct much of the image.

SECTION 3: VECTORS

See included exercises with code download:

- linalg_matrices.pdf
- linalg_vectors.m
- linalg_vectors.ipynb

8. ALGEBRAIC AND GEOMETRIC INTERPRETATIONS OF VECTORS

Vector: An ordered list of numbers

Dimensionality: The number of elements

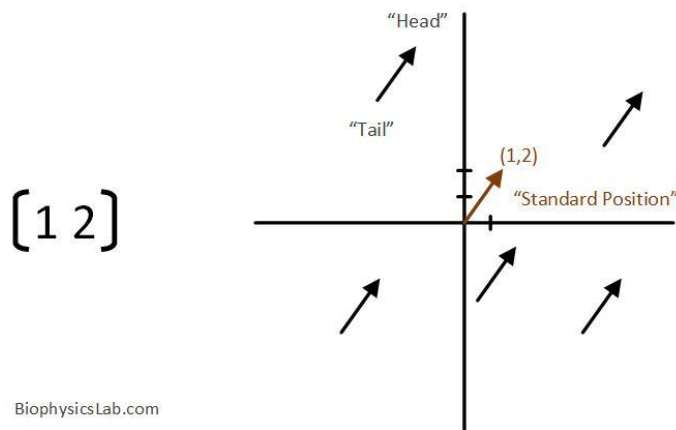
Example of two different 3D column vectors

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \neq \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

Vector symbol: \mathbf{v} or $\vec{\mathbf{v}}$ or $\vec{\mathbf{v}}$ or \boldsymbol{v}

Geometric vectors are robust to translation or rigid body motion

Two Dimensional Vector



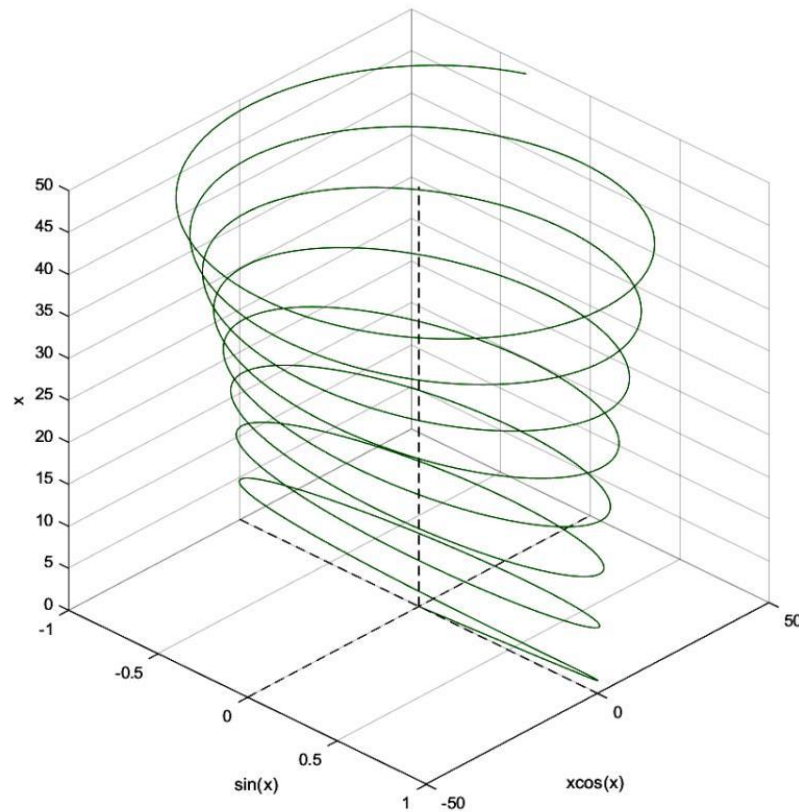
BiophysicsLab.com

Figure 3 - Two-dimensional vector

Note: Vector $\mathbf{v} = \begin{bmatrix} 1 & 2 \end{bmatrix}$ is not the same as point = (1,2)

Vector-valued Functions

$$\begin{pmatrix} \sin(x) \\ x\cos(x) \\ x \end{pmatrix}$$



BiophysicsLab.com

Figure 4 - Vector valued functions

```
%% Vector-valued Functions: Distorted Helix
% data
x = 0:pi/50:50;
v3(:,1) = sin(x);
v3(:,2) = x.*cos(x);
v3(:,3) = x;

% the main plotting code:
plot3(v3(:,1),v3(:,2),v3(:,3), 'color', [0 0.5 0]) % dark green color

% make the plot look nicer
axis square
axis([-1 1 -50 50 0 50])
hold on, grid on
plot3(get(gca,'xlim'),[0 0],[0 0],'k--')
plot3([0 0],get(gca,'ylim'),[0 0],'k--')
plot3([0 0],[0 0],get(gca,'zlim'),'k--')
xlabel('sin(x)')
ylabel('xcos(x)')
zlabel('x')
```

```
% might be easier to see when rotated  
rotate3d on
```

MATLAB

Code: linalg_vectors.m

Lines 9 to 59

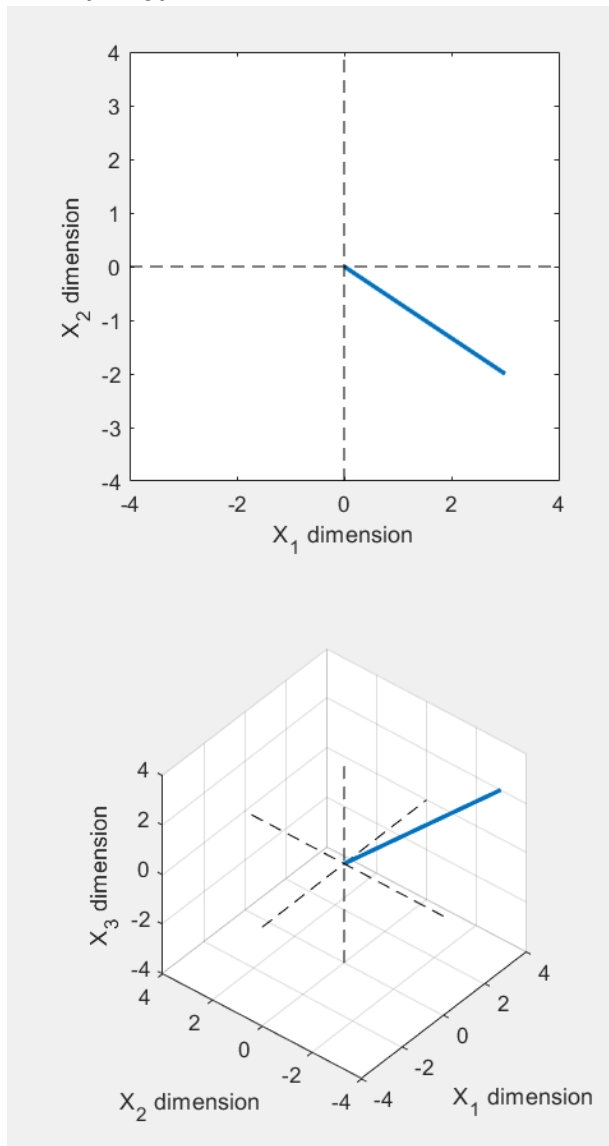


Figure 5 – MATLAB vector plot in 2D and 3D demonstration

9. VECTOR ADDITION AND SUBTRACTION

Vector addition: Geometry

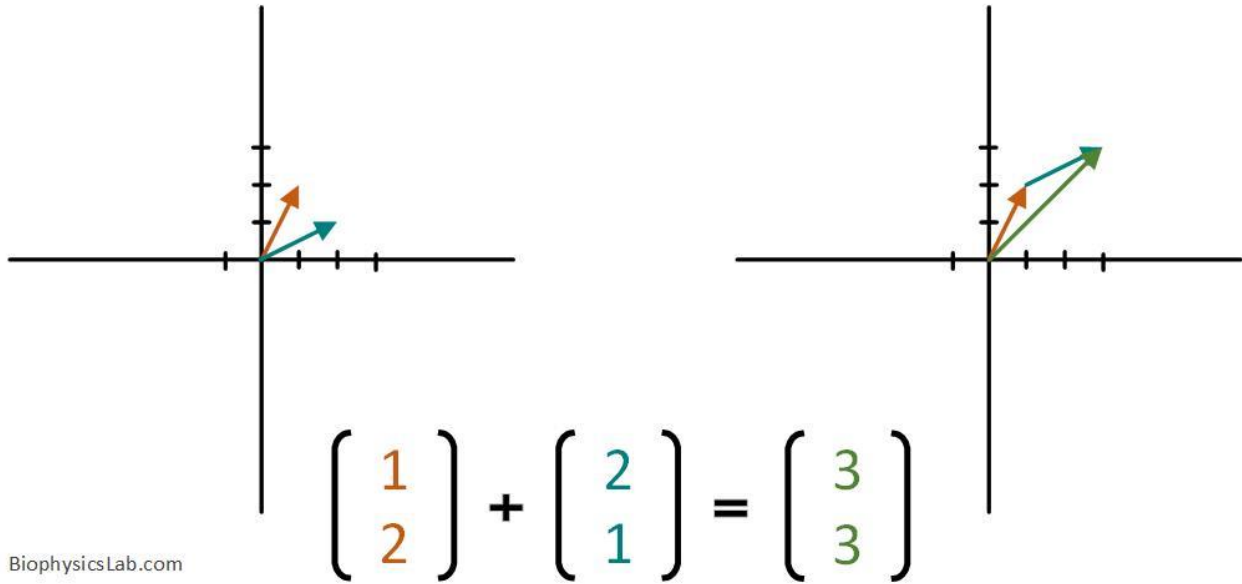


Figure 6 - Vector addition: Geometry

Vector translation: Move a vector from standard position, such as in example above.

Vector subtraction:

Vector Subtraction: Geometry

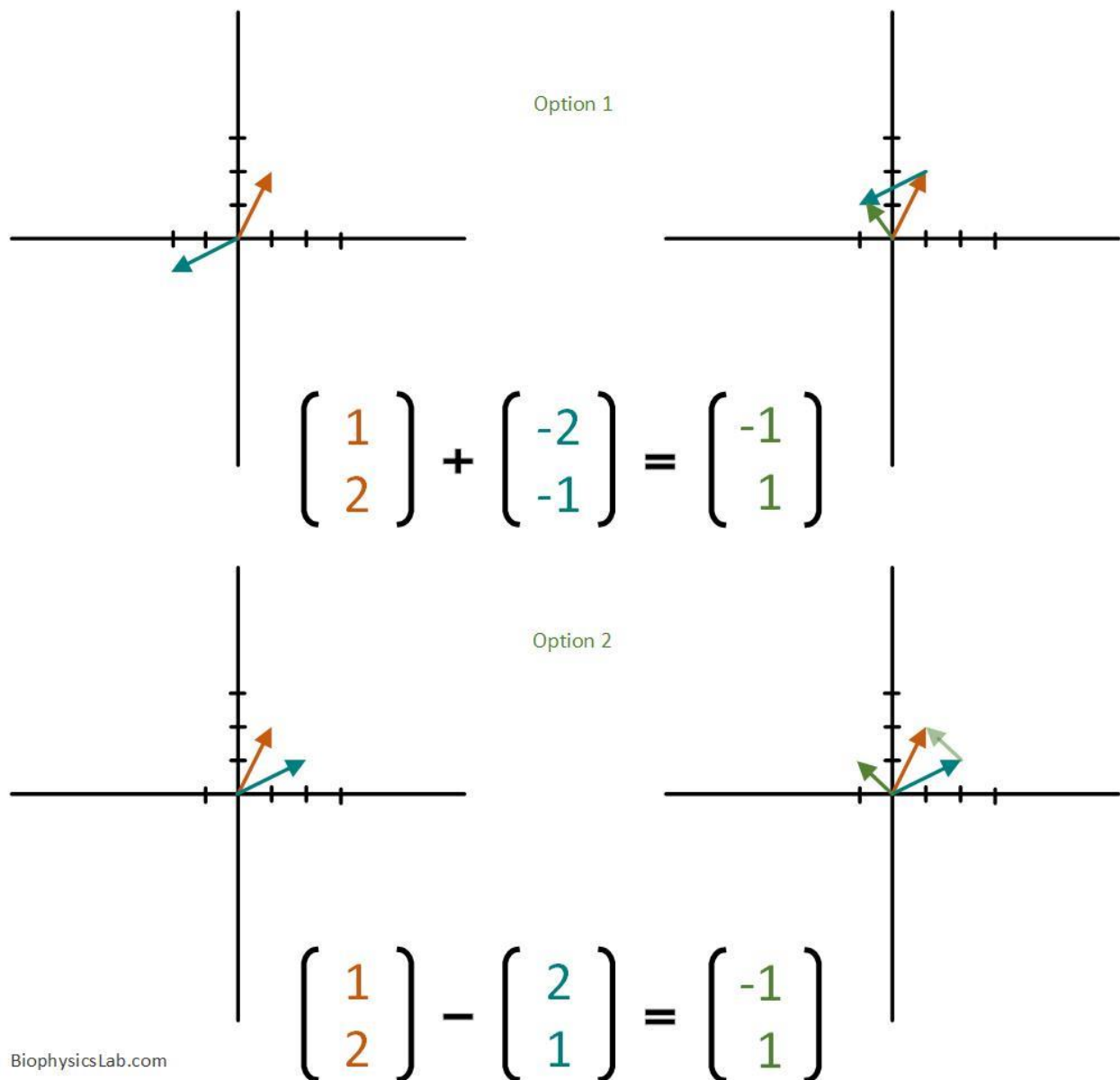


Figure 7 - Vector subtraction: Geometry (2 methods)

Option 1: multiply second vector by -1 (change sign of each element), then as in addition example, translate second vector head to tail of first vector, finally draw resulting vector from head of first vector to tail of second vector.

Option 2: with both vectors in the standard position, draw resulting vector from head of second vector to head of first vector.

MATLAB

Code: linalg_vectors.m

Lines 60 to 93

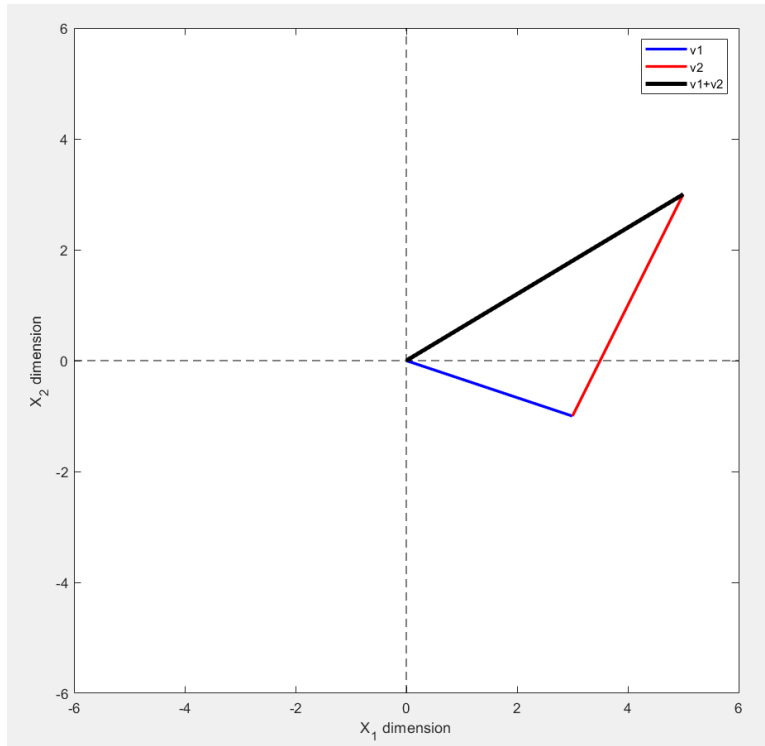


Figure 8 - MATLAB vector addition demonstration

10. VECTOR-SCALAR MULTIPLICATION

Table 1 - Linear algebra terminology

Object	$\begin{pmatrix} -1 & 0 & 2 & 8 \\ 0 & 1 & 4 & 4 \\ 1 & 4 & 9 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$	7
Name	“Matrix”	“Vector”	“Scalar”
Symbol	A, M, C	v, u, w	α, β, γ

Vector scalar multiplication: changes length of vector but not direction.

Eigenvalue decomposition is based on this concept of scalar vector multiplication not changing which linear subspace the original vector lies on.

MATLAB

Code: linalg_vectors.m

Lines 94 to 124

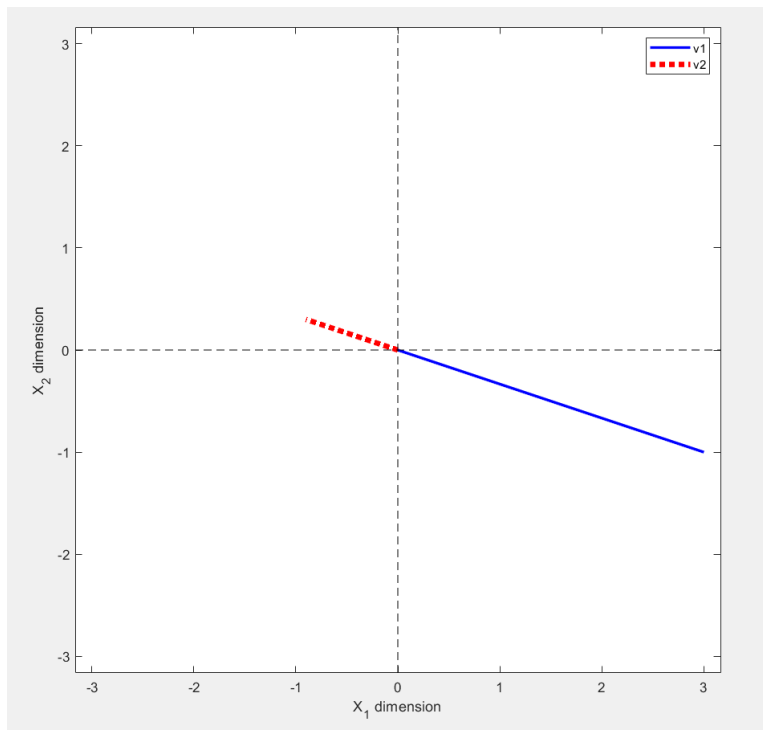


Figure 9 – MATLAB vector-scalar multiplication demonstration

11. VECTOR - VECTOR MULTIPLICATION: THE DOT PRODUCT

Dot product: a single number which is the sum of point-wise multiplication of two vectors (a.k.a. the scalar product or inner product).

Algorithms that use the dot product as basic building blocks:

- Correlation
- Convolution
- Fourier transform

Definition of the dot product:

$$\alpha = \mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T \mathbf{b} = \sum_{i=1}^n \mathbf{a}_i \mathbf{b}_i$$

Dot product example:

Dot product (a.k.a. scalar product) example

$$\begin{array}{c} \mathbf{v} \\ \left(\begin{array}{c} 1 \\ 0 \\ 2 \\ 5 \\ -2 \end{array} \right) \end{array} \quad \begin{array}{c} \mathbf{w} \\ \left(\begin{array}{c} 2 \\ 8 \\ -6 \\ 1 \\ 0 \end{array} \right) \end{array} \quad \begin{array}{l} \mathbf{v}^T \mathbf{w} = 1*2 + 0*8 + 2*(-6) + 5*1 + (-2)*0 \\ = 2 - 12 + 5 \\ = -5 \end{array}$$

BiophysicsLab.com

Figure 10 - Dot product example

MATLAB

Code: linalg_vectors.m

Lines 125 to 158

Four methods to compute dot product in MATLAB

sum(v1.*v2)=32, dot(v1,v2)=32, v1*v2'=32, for loop:32

12. DOT PRODUCT PROPERTIES: ASSOCIATIVE, DISTRIBUTIVE, COMMUTATIVE

Distributive property for scalar

$$a(b + c) = ab + ac$$

Distributive property for vector dot products

$$\mathbf{a}^T (\mathbf{b} + \mathbf{c}) = \mathbf{a}^T \mathbf{b} + \mathbf{a}^T \mathbf{c}$$

Proof for distributive property for vector dot products

$$\sum_{i=1}^n a_i b_i + \sum_{i=1}^n a_i c_i = \sum_{i=1}^n a_i (b_i + c_i)$$

Associative property for scalar

$$a(b \cdot c) = (a \cdot b)c$$

Associative property for vector dot products

$$\mathbf{a}^T (\mathbf{b}^T \mathbf{c}) \neq (\mathbf{a}^T \mathbf{b})^T \mathbf{c}$$

Scalar

Scalar

Proof for associative property for vector dot products

Given

$$\mathbf{a} \in \mathbb{R}^5$$

$$\mathbf{b}, \mathbf{c} \in \mathbb{R}^3$$

Then

$$\mathbf{a}^T (\mathbf{b}^T \mathbf{c}) \quad (\mathbf{a}^T \mathbf{b})^T \mathbf{c}$$



Yet Matrix multiplication is associative

$$\mathbf{A}(\mathbf{B}\mathbf{C}) = (\mathbf{A}\mathbf{B})\mathbf{C}$$

MATLAB

linalg_vectors.m

Lines 159 to 200

```
% Distributive example
res1 = a' * (b+c);
res2 = a'*b + a'*c;

% Associative example
res1 = a' * (b'*c);
res2 = (a'*b)' * c;

% Special cases where associative property works!
% 1) one vector is the zeros vector
% 2) a==b==c
```

13. CODE CHALLENGE: DOT PRODUCTS WITH MATRIX COLUMNS

MATLAB

Lesson_13_Dot_Product_Mat.m

```
% Create 2 4x6 matrices of random numbers.
A = randn(4,6);
B = randn(4,6);

% Use a for-loop to compute dot products between corresponding columns.
dpw = zeros(size(A,2),1);
for i=1:size(A,2)
    dps(i) = dot( A(:,i),B(:,i) );
end
disp(dps);
```

14. VECTOR LENGTH

Vector length is also called the magnitude or norm

Vector length via geometry (and Pythagoras)

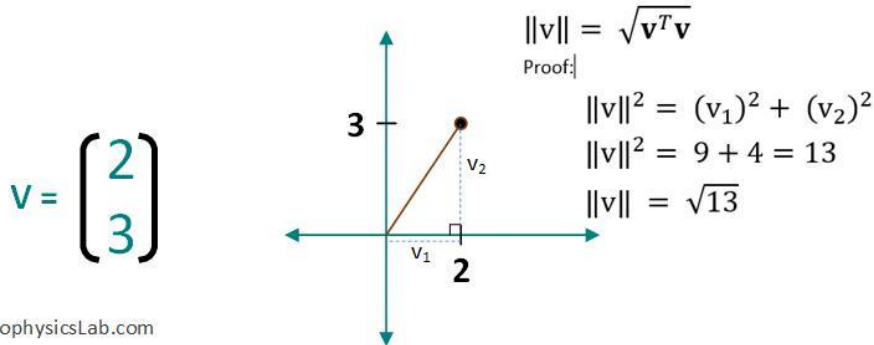


Figure 11 - Vector length via geometry (and via Pythagoras)

MATLAB

linalg_vectors.m

Lines 201 to 216

```
v1 = [ 1 2 3 4 5 6 ];  
  
% methods 1-4, just like with the regular dot product, e.g.:  
vl = sqrt( sum(v1.*v1) );  
  
% method 5: take the norm  
vl = norm(v1);
```

15. DOT PRODUCT GEOMETRY: SIGN AND ORTHOGONALITY

UNDERSTAND THE GEOMETRIC INTERPRETATION OF THE DOT PRODUCT

Acute angle $< 90^\circ$, $\cos(\Theta) > 0$, α (dot product) > 0

Obtuse angle $> 90^\circ$, $\cos(\Theta) < 0$, $\alpha < 0$

Orthogonal angle (perpendicular) $= 90^\circ$, $\cos(\Theta) = 0$, $\alpha = 0$

Zero angle $= 0^\circ$, $\cos(\Theta) = 1$, $\alpha = |\mathbf{a}||\mathbf{b}|$

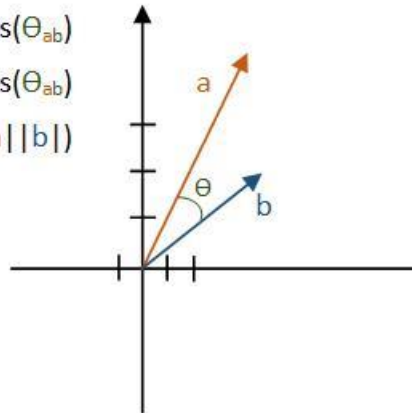
KNOW HOW TO COMPUTE THE ANGEL BETWEEN ANY TWO VECTORS

How to compute the angle between any two vectors

Dot product: $\alpha = |\mathbf{a}| |\mathbf{b}| \cos(\theta_{ab})$

$$\alpha / |\mathbf{a}| |\mathbf{b}| = \cos(\theta_{ab})$$

$$\theta_{ab} = \arccos(\alpha / |\mathbf{a}| |\mathbf{b}|)$$



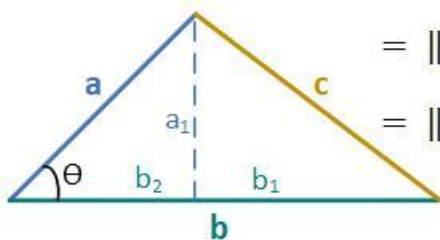
BiophysicsLab.com

Figure 12 - How to compute the angle between any two vectors

SEE THE EQUIVALENCE OF THE ALGEBRAIC AND GEOMETRIC PERSPECTIVES ON THE DOT PRODUCT

Vector algebra vs. geometry: They're the same

$$\|\mathbf{c}\|^2 = \|\mathbf{a} - \mathbf{b}\|^2 = (\mathbf{a} - \mathbf{b})^T (\mathbf{a} - \mathbf{b})$$



$$= \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\mathbf{a}^T \mathbf{b}$$

$$= \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{a}\| \|\mathbf{b}\| \cos(\theta_{ab})$$

BiophysicsLab.com

Figure 13 - Vector algebra vs. geometry

MATLAB

linalg_vectors.m

Lines 217 to 261

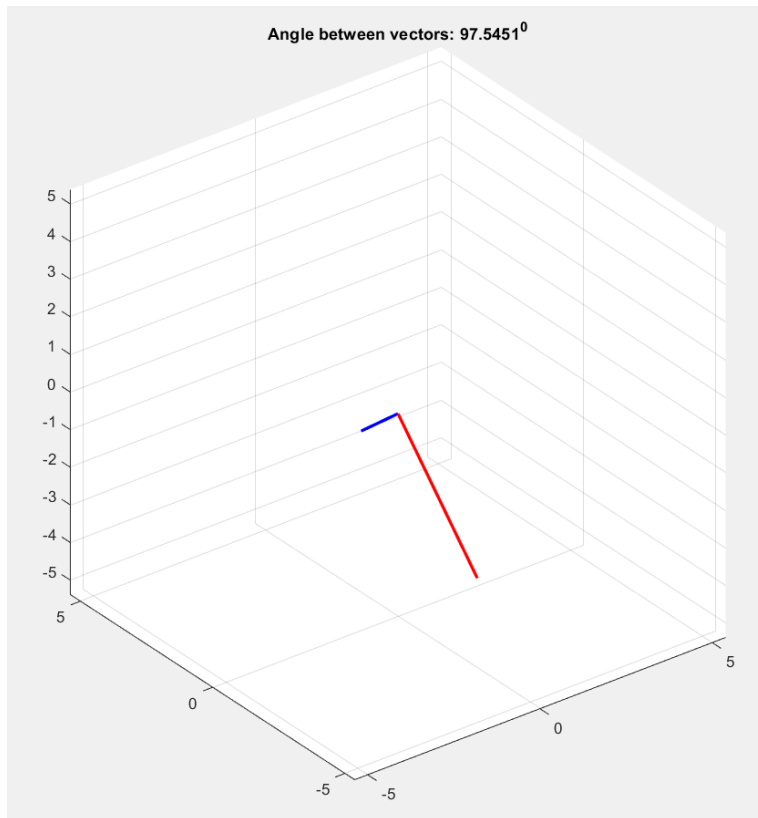


Figure 14 - MATLAB dot product demonstration

```
% compute the angle (radians) between two vectors
ang = acos( dot(v1,v2) / (norm(v1)*norm(v2)) );

% algebraic
dp_a = sum( v1.*v2 );

% geometric
dp_g = norm(v1)*norm(v2)*cos(ang);

% print dot product to command
disp([ 'Algebra: ' num2str(dp_a) ', geometry: ' num2str(dp_g) ])
Algebra: -3, geometry: -3
```

16. CODE CHALLENGE: DOT PRODUCT SIGN AND SCALAR MULTIPLICATION

Soft proof: verify a coded script works

MATLAB

Lesson 16 Code Challenge Dot Product with Scalars.m

```
%% Test whether the dot product sign is invariant to scalar multiplication
% In general, the dot product is not invariant to scalar multiplication,
```

```
% except when the vectors are Orthogonal (perpendicular).
```

```
% Generate two vectors (R3)
```

```
% v1 = [-3 4 5]'; % Orthogonal
```

```
v1 = [-3 4 6]'; % -3 Obtuse
```

```
v2 = [ 3 6 -3]';
```

```
% Generate two scalars
```

```
% s1 = 2;
```

```
s1 = -2; % flip v1
```

```
s2 = 3;
```

```
% Compute the dot product between vectors
```

```
disp(' ');
```

```
disp(['original: ' num2str(dot(v1,v2)) ])
```

```
% Compute the dot product between the scaled vectors
```

```
disp(['scaled: ' num2str( (s1*v1)' * (s2 * v2) ) ])
```

17. CODE CHALLENGE: CAUCHY-SCHWARZ INEQUALITY

Prove the C-S inequality

The Cauchy-Schwarz inequality

$$|\mathbf{a}^T \mathbf{b}| \leq \|\mathbf{a}\| \|\mathbf{b}\|$$

$$\|\mathbf{a}\| \|\mathbf{b}\| |\cos(\theta_{ab})| \leq \|\mathbf{a}\| \|\mathbf{b}\|$$

$\underbrace{\hspace{10em}}$

$$0 \leq |\cos(\theta_{ab})| \leq 1$$

BiophysicsLab.com

Figure 15 - The Cauchy-Schwarz inequality

Determine the conditions for equality

- Left hand side of equation will be smaller than the right-hand side when the angle between vectors **a** and **b** is anything other than 0° and 180° (π radians).
- If the vectors **a** and **b** form a linearly independent set then left-hand side of the equations will be smaller than the right-hand side of the equation.
- If the vectors **a** and **b** form a linearly dependent set then the one vector is a scaled version of the other vector then we have an equality.

MATLAB

Lesson 17 New Code Challenge Cauchy Schwarz inequality.m

```
% Create three vectors, one of which is dependent on the other
a = randn(5,1);
b = randn(5,1);
c = randn*a;

% Dot products (LHS of the equation in lecture)
aTb = dot(a,b);
aTc = dot(a,c);

clc
% expect term 1 to be smaller than term 2
disp([ abs(aTb) norm(a)*norm(b) ])
% expect term 1 to be same as term 2
disp([ abs(aTc) norm(a)*norm(c) ])
```

18. CODE CHALLENGE: IS THE DOT PRODUCT COMMUTATIVE?

MATLAB

Lesson 16 Code Challenge Dot Product Commutative.m

```
%% Is the dot product commutative?
% Answer is yes, since dot product uses scalar math.
% a'*b == b'*a

% 1) generate two 100-element random row vectors, compute dot product a with b, b with a
a = randn(1,100);
b = randn(1,100);

% dot product is a row vector * a column vector
disp([ a*b' b*a' ]) % values are the same

% 2) generate two 2-element integer row vectors, repeat
a = [2 4];
b = [3 5];

disp([ a*b' b*a' ]) % values are the same
```

19. VECTOR HADAMARD MULTIPLICATION

Also known as element-wise multiplication.

Hadamard (element-wise) multiplication

$$\begin{pmatrix} -1 \\ 0 \\ 2 \\ 5 \\ -2 \end{pmatrix} \odot \begin{pmatrix} 2 \\ 8 \\ -6 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -12 \\ 5 \\ 0 \end{pmatrix}$$

BiophysicsLab.com

Figure 16 - Hadamard multiplication

Note: See 55. Hadamard (element-wise) multiplication

MATLAB

linalg_vectors.m

Lines 262 to 275

```
w1 = [ 1 3 5 ];  
w2 = [ 3 4 2 ];  
  
w3 = w1 .* w2
```

20. OUTER PRODUCT

Dot product: $\underset{N \times 1}{\mathbf{V}}^T \underset{N \times 1}{\mathbf{W}} = 1 \times 1$

Outer product: $\underset{N \times 1}{\mathbf{V}} \underset{M \times 1}{\mathbf{W}}^T = N \times M$

Outer product

$$\begin{pmatrix} 1 \\ 0 \\ 2 \\ 5 \end{pmatrix} \begin{bmatrix} a & b & c & d \end{bmatrix} = \begin{pmatrix} 1a & 1b & 1c & 1d \\ 0a & 0b & 0c & 0d \\ 2a & 2b & 2c & 2d \\ 5a & 5b & 5c & 5d \end{pmatrix}$$

BiophysicsLab.com

Figure 17 - Outer product

MATLAB

linalg_vectors.m

Lines 276 to 301

```
v1 = [ 1 2 3 ]'; % transpose (') to make column vectors
v2 = [ -1 0 ]';

% dot (inner) product! Will produce size error
v1'*v2

% outer product
v1*v2'

% terrible programming, but helps conceptually:
op = zeros(length(v1),length(v2));
for i=1:length(v1)
    for j=1:length(v2)
        op(i,j) = v1(i) * v2(j);
    end
end
```

21. VECTOR CROSS PRODUCT

Defined only for two 3D vectors.

Result is another 3D vector.

Cross-product: algebraic formula

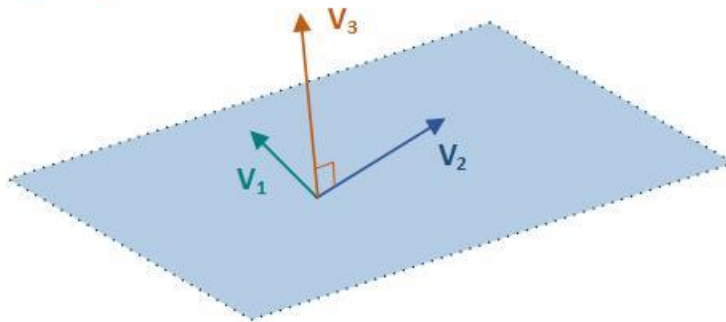
$$\begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} \times \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 2c - 3b \\ 3a - 1c \\ 1b - 2a \end{pmatrix}$$

BiophysicsLab.com

Figure 18 - Cross-product: algebraic formula

Cross-product: Geometric formula

$$\mathbf{V}_1 \times \mathbf{V}_2 = \mathbf{V}_3$$



BiophysicsLab.com

Figure 19 Cross-product: Geometric formula

MATLAB

linalg_vectors.m

Lines 302 to 339

Keywords: cross, ezmesh, plot3

```
% create vectors  
v1 = [-3 2 5];
```

```

v2 = [ 4 -3 0 ];

% MATLAB's cross-product function
v3a = cross( v1,v2 );

% "manual" method
v3b = [ v1(2)*v2(3) - v1(3)*v2(2) ;
        v1(3)*v2(1) - v1(1)*v2(3) ;
        v1(1)*v2(2) - v1(2)*v2(1) ];

```

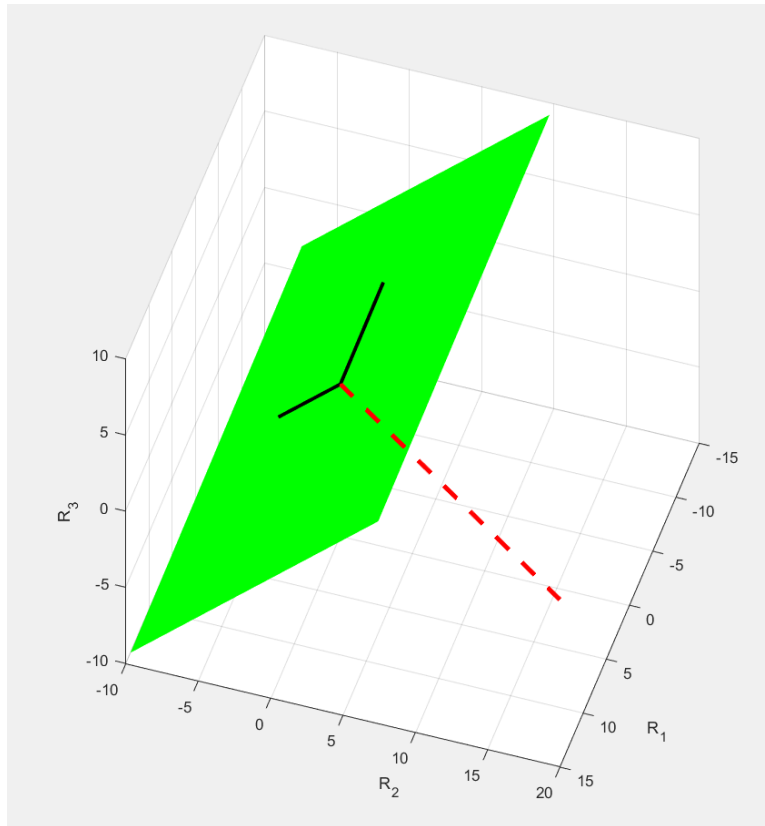


Figure 20 - MATLAB 3D plot of vector cross product

22. VECTORS WITH COMPLEX NUMBERS

$$z = a + bi \in \mathbb{C}$$

Where

$$i = \sqrt{-1}$$

Vector with real and imaginary dimensions

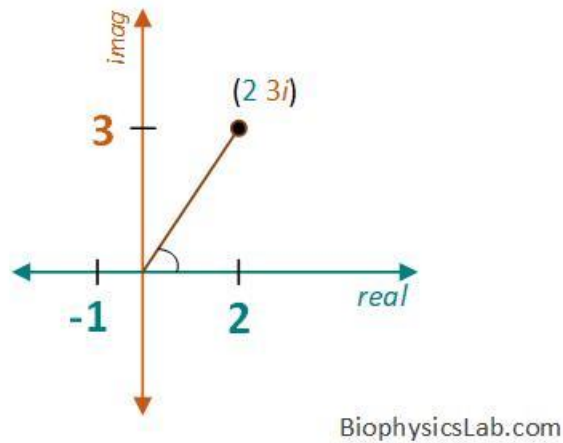


Figure 21 - Vector with real and imaginary dimensions

Dot product with complex vectors example

$$\begin{pmatrix} 1 & 3i \\ -2i & 4 \\ -5 & \end{pmatrix}^T \begin{pmatrix} 6 & 2i \\ 8 & 3i \\ -5 & \end{pmatrix} = (1 \ 3i)(6 \ 2i) + -16i + 12i + 25 \\
 = 6 + 2i + 18i - 6 - 16i + 12i + 25 \\
 = 25 + 16i$$

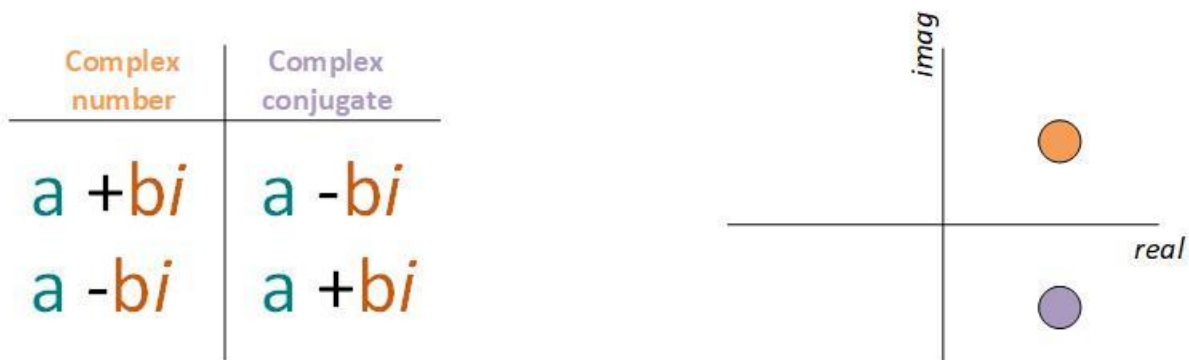
BiophysicsLab.com

Figure 22 - Dot product with complex vectors example

23. HERMITIAN TRANSPOSE (A.K.A. CONJUGATE TRANSPOSE)

The complex conjugate of a complex number is simply the complex number with sign change associated with the complex number. The real number in a complex number does not change. For real number vectors or matrices, the Hermitian transpose (z'^*z) is the same as the regular transpose ($z.'*z$).

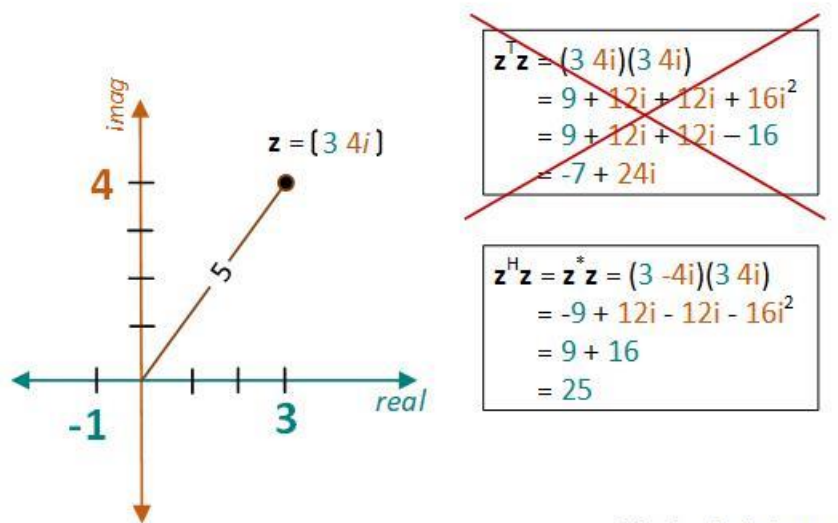
Hermitian transpose



BiophysicsLab.com

Figure 23 - Hermitian transpose

Hermitian transpose example: Length of vector



BiophysicsLab.com

Figure 24 - Hermitian transpose example: Length of vector

MATLAB

linalg_vectors.m

Lines 340 to 370

Keywords: complex, norm, transpose, ctranpose

```
% create a complex number
z = complex(3,4);

% magnitude
norm(z)

% by transpose?
transpose(z)*z

% by Hermitian transpose
z'*z

% not the Hermitian:
z.'*z
```

24. INTERPRETING AND CREATING UNIT VECTORS

Unit vector – a length of one.

Zero vector has no unit vector.

To find a scalar to convert a vector into a unit vector of length 1, multiply the vector by the reciprocal of the length (norm) of the vector.

Unit vectors don't change the magnitude of a vector, but can change the direction of the vector.

MATLAB

linalg_vectors.m

Lines 371 to 403

Keywords: [] (vector) norm, plot, hold (plot), 'linew' (plot), 'AutoUpdate' (legend), legend (plot), axis (plot), xlabel (plot), ylabel (plot)

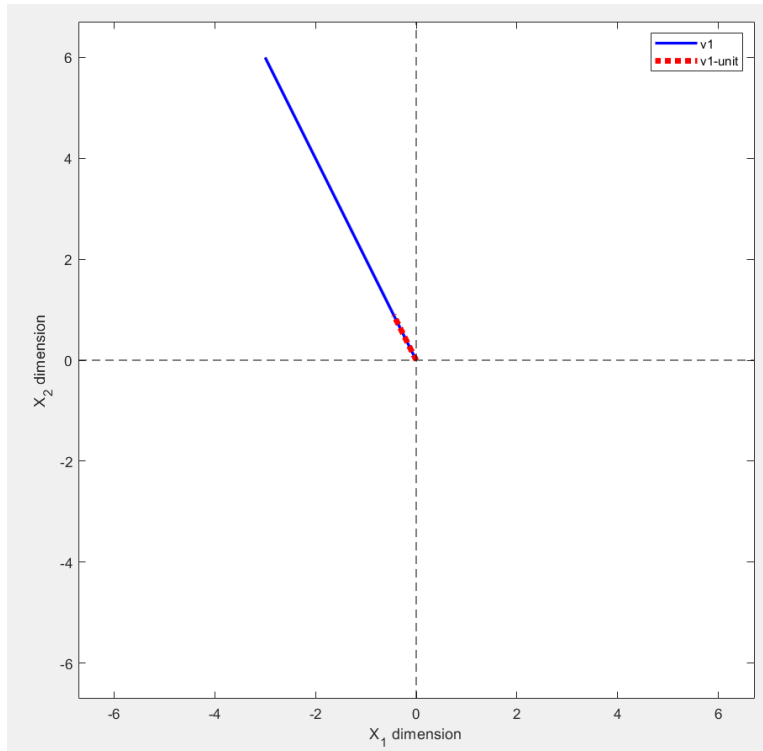


Figure 25 - MATLAB unit vector example

25. CODE CHALLENGE: DOT PRODUCTS WITH UNIT VECTORS

MATLAB

Lesson 24 Code Challenge Dot Products with Vectors.m

```
% Compute the lengths of the individual vectors, and magnitude of their dot
% product.

% 'Normalize' the vectors (create unit vectors along the same direction).

% Compute the magnitude of that dot product:
% Used in statistics: Pearson Correlation coefficient
% Used in machine learning: Cosine similarity
```

26. DIMENSIONS AND FIELDS IN LINEAR ALGEBRA

Dimension: number of elements in a vector where each number corresponds to a new direction (or subspace) like x, y, z in 3D.

Field: A set (typically, numbers) on which addition, subtraction, multiplication, and division are valid operators.

Using hollow letters:

- \mathbb{R} Real numbers (the familiar number line)
- \mathbb{C} Complex numbers (a *i*b)

- \mathbb{Z} Integers (“counting numbers”) – not a field

Example:

$$\mathbf{v} \in \mathbb{R}^N$$

Vector \mathbf{v} is a member of the N dimensional field \mathbb{R}^N

27. SUBSPACES

A subspace is defined as a set of all vectors that can be created by taking linear combinations of some vector or a set of vectors. Where linear combination means multiplying a vector by a scalar. Also combining two vectors with their associated scalar multipliers also forms a subspace.

$$\lambda \mathbf{v}, \quad \lambda \in \mathbb{R}$$

$$\lambda \mathbf{v} + \beta \mathbf{w}, \quad \lambda, \beta \in \mathbb{R}$$

A vector subspace must:

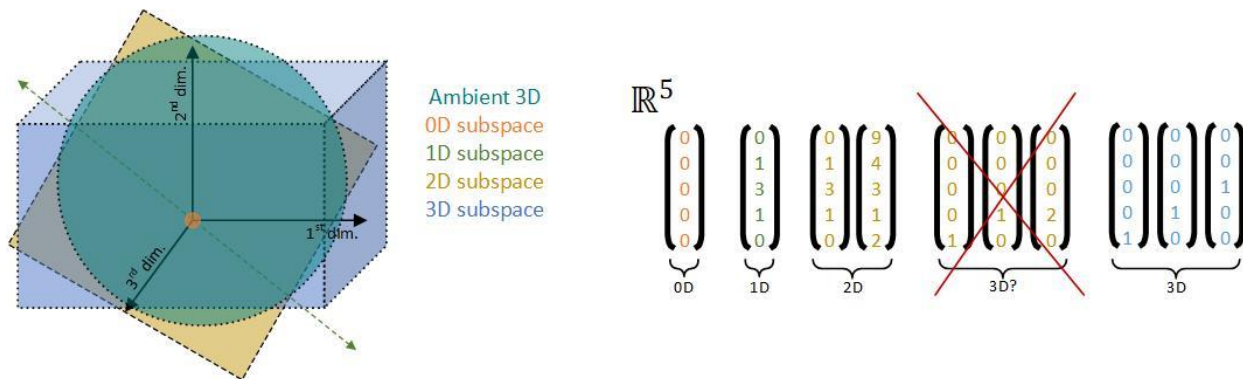
- Be closed under addition and scalar multiplication,
 - $\forall \mathbf{v}, \mathbf{w} \in V; \quad \forall \lambda, \alpha \in \mathbb{R}; \quad \lambda \mathbf{v} + \alpha \mathbf{w} \in V$
 - Above set of equations reads: For any vectors \mathbf{v} and \mathbf{w} that are both contained in the subspace V , and for any scalars α and λ that are contained in the set of real numbers \mathbb{R} , then $\lambda \mathbf{v}$ plus $\alpha \mathbf{w}$ must also be contained within the same subspace V .
- Contain the zero vector.
 - True since scalars α and λ could be zero as zero is also part of \mathbb{R} .

All subspaces intersect at the origin.

Two vectors and all their linear combinations form a 2D plane subspace so long as both vectors are not multiples of each other.

There is one zero-dimensional subspace – the zero vector

Ambient spaces and subspaces: Geometry and Algebra



BiophysicsLab.com

Figure 26 - Ambient spaces and subspaces: Geometry and Algebra

28. SUBSPACES VS. SUBSETS

Subset: a set of points that satisfies some conditions:

- Doesn't need to include the origin.
- Doesn't need to be closed under addition or multiplication.
- Can have boundaries.

Subset examples:

- All points on the XY plane such that $x > 0, y > 0$
- All points on the XY plane such that $x^2 + y^2 = 1$ (a circle)
- All points on the XY plane such that $y = 4x$, for all x (a line) – Both a subset and a subspace
- All points on the XY plane such that $y = 4x + 1$, for all x (a line) – A subset only (does not include origin)

Given a set of criteria follow these steps to differentiate between a subspace and a subset:

1. Determine whether the origin is in the set.
2. Try to write down the criteria in terms of scalars and vectors of the form $\alpha \mathbf{v} + \beta \mathbf{w}$

29. SPAN

Subspace: a region of space that can be reached by any linear combination of the given vectors. Then those vectors *span* that subspace.

$$\text{span}(\{\mathbf{v}_1, \dots, \mathbf{v}_n\}) = \alpha_1 \mathbf{v}_1 + \dots + \alpha_n \mathbf{v}_n, \alpha \in \mathbb{R}$$

Span: algebraic example

$$\mathbf{v} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} \quad S = \left\{ \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 7 \\ 0 \end{pmatrix} \right\}$$

$$\mathbf{v} \in \text{span}(S) \text{ because } \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} = \frac{5}{6} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + \frac{1}{6} \begin{pmatrix} 1 \\ 7 \\ 0 \end{pmatrix}$$
$$\mathbf{w} \notin \text{span}(S)$$

BiophysicsLab.com

Figure 27 - Span: algebraic example

The weightings for span set S can be determined using methods to be described later in this class. The two vectors within S must be independent of each other. Vector \mathbf{w} is not contained within S because both vectors in S have a zero element in the z dimension making independent weights impossible to find.

MATLAB

linalg_vectors.m

Lines 404 to 437

Keywords: cross, surf, shading interp, axis square, grid on, rotate3d on

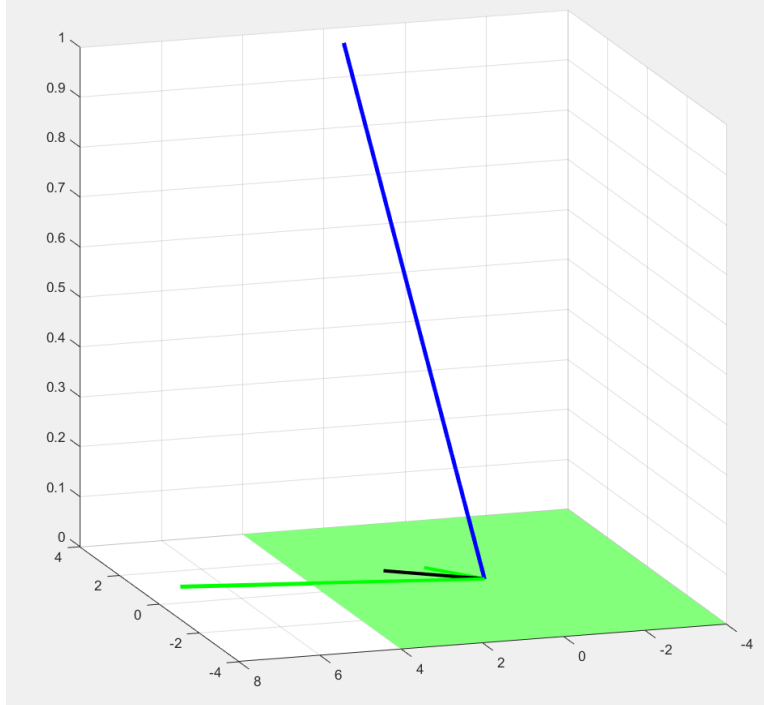
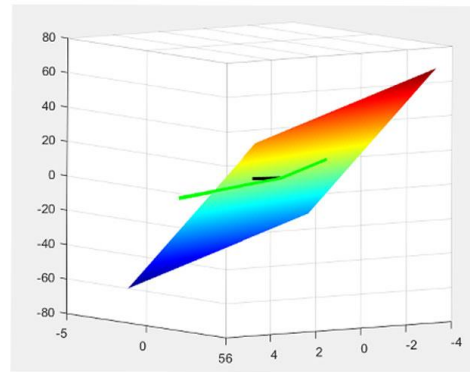
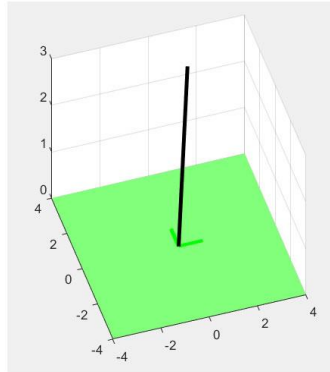
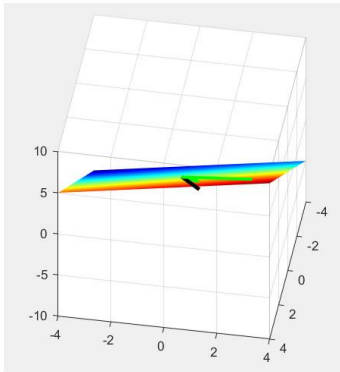


Figure 28 - MATLAB span demonstration

QUIZ 5: IN THE SPAN?

Lesson 28 Quiz: Which span contains vector $[1 \ 2 \ 3]$?



$\{ [1 \ 2 \ 4], [3 \ 2 \ 5] \}$ **X**

$\{ [1 \ 0 \ 0], [0 \ 1 \ 0] \}$ **X**

$\{ [2 \ 6 \ -3], [0 \ -2 \ 9] \}$ **✓**

Figure 29 Quiz 5: In the span?

Only one span contains the vector. As seen in the images, the correct span and vector are all in the same 2D plane.

MATLAB

Lesson_28_Quiz_5.m

30. LINEAR INDEPENDENCE

A set of M vectors is independent if each vector points in a geometric dimension not reachable using other vectors in the set.

Example of linearly dependent sets

$$\left\{ \mathbf{w}_1, \mathbf{w}_2 \right\} = \left\{ \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \right\} \quad \mathbf{w}_2 = 2\mathbf{w}_1$$

$$\left\{ \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \right\} = \left\{ \begin{pmatrix} 0 \\ 2 \\ 5 \end{pmatrix}, \begin{pmatrix} -27 \\ 5 \\ -37 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \\ 8 \end{pmatrix} \right\} \quad \mathbf{v}_2 = 7\mathbf{v}_1 - 9\mathbf{v}_3$$

BiophysicsLab.com

Figure 30 - Example of linearly dependent sets

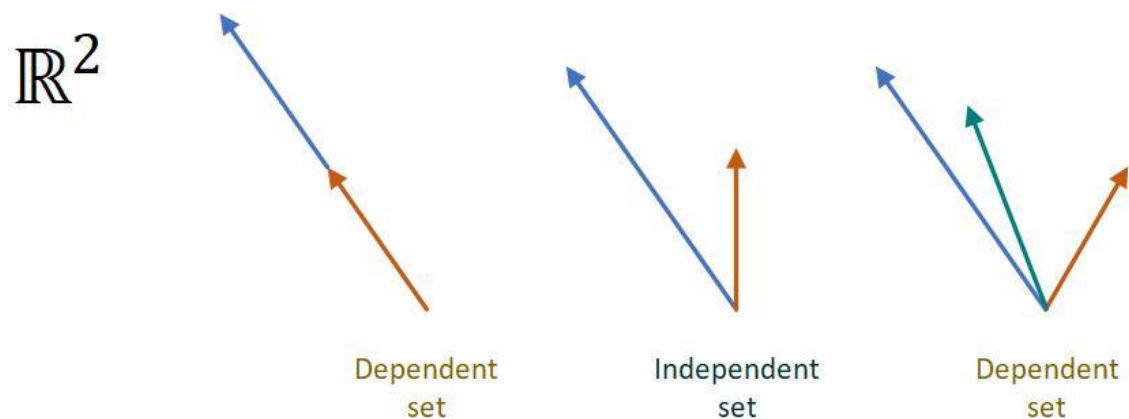
Linear dependence: formal definition

$$0 = \lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \cdots + \lambda_3 \mathbf{v}_3, \quad \lambda \in \mathbb{R}$$

Any set of $M > N$ vectors in \mathbb{R}^N is dependent.

Any set of $M \leq N$ vectors in \mathbb{R}^N could be independent.

Linear independence: geometry



BiophysicsLab.com

Figure 31 - Linear independence: geometry

Note in figure above: Any three vectors in 2D space will form a dependent set because any two vectors multiplied by the right non-zero scalars will form the third vector.

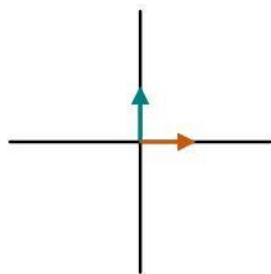
Determine whether a set is independent:

1. Count vectors and compare with \mathbb{R}^N .
2. Check for 0's in corresponding (or all) elements.
3. Educated guess and test.
4. Matrix rank method (to be discussed later in course).

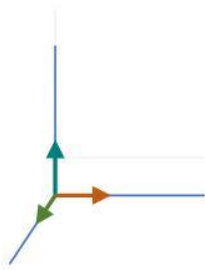
31. BASIS

Basis combines span and independence.

Example of standard basis vectors



$$\mathbb{R}^2 \quad \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$



$$\mathbb{R}^3 \quad \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$

BiophysicsLab.com

Figure 32 - Example of standard basis vectors

Each standard basis vector is of unit length and orthogonal to each other.

Each vector set form a basis because they contain a set of independent vectors that span the full subspace. Any point in \mathbb{R}^2 or \mathbb{R}^3 can be obtained by some unique linear combination of the standard basis vectors in that set.

A basis is like a ruler that measures distance and set of coordinates in some space.

When using a non-standard basis set, label points with a subscript in brackets for that basis: $P_{[T]}$

Two ways to compute an optimal set of basis vectors for a given data science or machine learning problem: “Principle Components Analysis“, “Independent Component Analysis”.

SECTION 4: INTRODUCTION TO MATRICES

See included exercises with code download:

- linalg_matrices.pdf
- linalg_matrices.m
- linalg_matrices.ipynb

33: MATRIX TERMINOLOGY AND DIMENSIONALITY

Matrix variable: Capital bold face letter

Elements within a matrix use a lower-case letter with row and column subscripted and separated by comma.

Block matrix: a block matrix or a partitioned matrix is a matrix that is interpreted as having been broken into sections called blocks or submatrices.

Matrix diagonal and off-diagonal elements

Reference elements in a matrix using rows M (first) and columns N (second). Mnemonic “MR NiCe guy”

Matrix dimensionality: number of elements in a matrix \mathbb{R}^{MN} (not the same as $\mathbb{R}^{M \times N}$), or a collection of column vectors where $C(M)$ in \mathbb{R}^M , or a collection of row vectors where $\mathbb{R}(M)$ in \mathbb{R}^N

Matrix is $M \times N$, while Tensor is $M \times N \times K$

Tensor used for storing data, or in physics can represent forces acting on an object as examples.

34. A ZOO OF MATRICES

Square: $M \times M$

Rectangular: $M \times N$ ($M \neq N$)

Symmetric: A square matrix with elements mirrored across the diagonal.

Skew-symmetric: Similar to symmetric matrix except the sign is flipped on mirrored elements - also the diagonal elements have to be all zeros.

Identity: All ones on the diagonal and all zero's everywhere else. Equivalent to the number 1:

Identity matrix is also square and symmetric. Often the symbol is a capital letter I with subscript describing the number of Rows/Columns. For example, I_4 representing an identity matrix with 4 rows and columns.

Zero: All elements are zero. Represented by a bold face $\mathbf{0}$ symbol.

Diagonal: All zeros on the off-diagonal elements. Matrix can be square or rectangular. If all diagonal elements are the same number, then the matrix can be represented by a scalar multiplied by the identity matrix: $\lambda \mathbf{I}$

Triangular: Two forms: upper triangular and lower triangular. Upper triangular matrix has numbers on and above the diagonal with all zeros below. Lower triangular matrix has numbers on and below the diagonal with all zeros above.

Augmented: Also called concatenated. Two matrices with the same number of rows where the right matrix is simply appended to the left matrix to make one larger matrix. Result has the same number of rows but now has more columns based on the appended number of rows in second matrix. One symbol to represent the catenation process is a “square U shape”. Also, common to

include a vertical line in the concatenated matrix showing the boundary where the second matrix was appended.

Complex: Complex matrix is one that has at least one element with a complex value number.

MATLAB

Code: linalg_matrices.m

Lines 1 to 33

Keywords: randn, eye, zeros, diag, triu, tril

```
%%  
% COURSE: Linear algebra: theory and implementation  
% SECTION: Introduction to matrices  
% VIDEO: A zoo of matrices  
% Instructor: sincxpress.com  
%  
%%  
  
% square vs. rectangular  
S = randn(5);  
S = randn(5,5);  
R = randn(5,2); % 5 rows, 2 columns  
  
% identity  
I = eye(3);  
  
% zeros  
Z = zeros(4);  
  
% diagonal  
D = diag([ 1 2 3 5 2 ]);  
  
% create triangular matrix from full matrices  
S = randn(5);  
U = triu(S);  
L = tril(S);  
  
% concatenate matrices (sizes must match!)  
A = randn(4,2);  
B = randn(4,4);  
C = [ A B ];  
  
%%
```

35. MATRIX ADDITION AND SUBTRACTION

Addition and subtraction is only defined for matrices with the same number of elements.

Matrix addition is commutative and associative.

“Shifting” a matrix away from degeneracy

$$\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} + 3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 7 \end{pmatrix}$$

$$\mathbf{A} + \lambda \mathbf{I} = \mathbf{C}$$

BiophysicsLab.com

Figure 33 - "Shifting" a matrix away from degeneracy

Shifting a matrix away from degeneracy. Regularize a matrix for machine learning. Used in Eigen decomposition and SVD.

MATLAB

Code: linalg_matrices.m

Lines 35 to 62

Keywords: randn, eye, shifting a matrix

```
%%  
%   COURSE: Linear algebra: theory and implementation  
%   SECTION: Introduction to matrices  
%   VIDEO: Matrix addition and subtraction  
%   Instructor: sincxpress.com  
%  
%%  
  
% create random matrices  
A = randn(5,4);  
B = randn(5,4);  
C = randn(5,4);  
  
% try to add them  
A+B  
A+C  
  
% "shifting" a matrix
```

```

l = .3; % lambda
N = 5; % size of square matrix
D = randn(N); % can only shift a square matrix

%
Ds = D + l*eye(N);

%%

```

36. MATRIX-SCALAR MULTIPLICATION

Scalar multiplication is carried out element-wise.
Pre- and post-multiplication are the same.

MATLAB

Code: linalg_matrices.m

Lines 64 to 80

Keywords: Matrix scalar multiplication

```

%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Introduction to matrices
% VIDEO: Matrix-scalar multiplication
% Instructor: sincxpress.com
%
%%

M = [1 2; 2 5];
s = 2;

% pre- and post-multiplication is the same:
M*s
s*M

%%

```

Result:

```

ans =
    2    4
    4   10

ans =
    2    4
    4   10

```

37. CODE CHALLENGE: IS MATRIX-SCALAR MULTIPLICATION A LINEAR OPERATION?

Test if scalar multiplication is linear by checking that scalar multiplication obeys the distributive property.

Test this equation: $s(A+B) = sA + sB$ where s is a scalar and A, B are $M \times N$ matrices

MATLAB

Code: Lesson_36_Code_Challenge_Matrix_Scalar.m

Answer: Yes, scalar multiplication is linear.

38. TRANSPOSE

Transpose a matrix: First column becomes the first row, etc...

Symmetric matrix: $A = A^T$

A square matrix, first column is the same as the first row, etc...

Skew-symmetric matrix: $A = -A^T$

A square matrix, first column is the same as the first row multiplied by -1, etc...

Complex numbers: "regular" transpose vs Hermitian transpose (see code in MATLAB section below)

MATLAB

Code: linalg_matrices.m

Lines 82 to 102

Keywords: Transpose, Hermitian transpose

```
%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Introduction to matrices
% VIDEO: Transpose
% Instructor: sincxpress.com
%
%%

M = [ 1 2 3; 2 3 4 ];

M'
M'' % note: " not "

% warning! be careful when using complex matrices
C = [ 4+1i 3 2-4i ];
C' % Hermitian transpose (complex numbers have sign flipped)
transpose(C) % "regular" transpose
C.' % "regular" transpose

%%
```

39. COMPLEX MATRICES

Complex matrix is one that has at least one element with a complex value number.

Used in signal processing: Fourier transform.

Hermitian transform is like a regular transform, but with complex value numbers sign changed.

Complex matrices

$$\begin{pmatrix} 1 & -1+5j & 0 \\ -1 & -2 & -4 \\ 6j & -4 & 5-2j \end{pmatrix}^H = \begin{pmatrix} 1 & -1 & -6j \\ -1-5j & -2 & -4 \\ 0 & -4 & 5+2j \end{pmatrix}$$

BiophysicsLab.com

Figure 34 - Complex matrices

40. DIAGONAL AND TRACE

Diagonal:

- Diagonal is defined for square and rectangular matrices.
- Diagonal of a covariance matrix contains the variance of each variable.
- Diagonal as a function takes a matrix as input and a vector as output.
- Diagonal is not the same as diagonalizing a matrix (used in Eigen decomposition).

$$v_i = \mathbf{A}_{i,i}, \quad i = \{1, 2, \dots, \min(m, n)\}$$

Diagonal

$$\text{diag} \left(\begin{pmatrix} 1 & -1 & 8 \\ -1 & -2 & 4 \\ 0 & 3 & 5 \end{pmatrix} \right) = \begin{pmatrix} 1 \\ -2 \\ 5 \end{pmatrix} \quad \text{diag} \left(\begin{pmatrix} 1 & -1 \\ -1 & -2 \\ 0 & 3 \end{pmatrix} \right) = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

BiophysicsLab.com

Figure 35 - Diagonal

Trace:

- Trace is defined for square matrices only.
- Trace is the sum of elements on the main diagonal (from the upper left to the lower right) of a square matrix.
- Trace as a function takes a matrix as input and a scalar as output.

$$tr(\mathbf{A}) = \sum_{i=1}^m \mathbf{A}_{i,i}$$

Trace

$$\text{trace} \left(\begin{pmatrix} 1 & -1 & 8 \\ -1 & -2 & 4 \\ 0 & 3 & 5 \end{pmatrix} \right) = 1 + (-2) + 5 = 4$$

BiophysicsLab.com

Figure 36 – Trace

MATLAB

Code: linalg_matrices.m

Lines 104 to 126

Keywords: round, randn, diag, trace, sum

```
%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Introduction to matrices
% VIDEO: Diagonal and trace
% Instructor: sincxpress.com
%
%%

% Generate 4x4 matrix of random integers
M = round( 5*randn(4) );

% extract the diagonals
d = diag(M);
```

```
% notice the two ways of using the diag function
d = diag(M); % input is matrix, output is vector
D = diag(d); % input is vector, output is matrix

% trace as sum of diagonal elements (calculate using 2 methods)
tr = trace(M);
tr2 = sum( diag(M) );

%% end.
```

41. CODE CHALLENGE: LINEARITY OF TRACE

1. Determine the relationship between $\text{tr}(A)+\text{tr}(B)$ and $\text{tr}(A+B)$. If they are equal then the relationship is linear.
2. Determine the relationship between $\text{tr}(g*A)$ and $g*\text{tr}(A)$ where g is a scalar. If they are equal then the relationship is linear.

MATLAB

Code: Lesson_40_Code_Challenge_Linearity_of_Trace.m

Answer; Yes, trace is a linear function using tests 1 and 2 above.

42. BROADCASTING MATRIX ARITHMETIC

“Broadcasting” is not valid in traditional linear algebra. But is used often in applied linear algebra, machine learning, deep learning.

Broadcasting works for all four basic arithmetic operations: $+$ $-$ $*$ $/$

Broadcasting means replicate a single row or column vector such that each row or column in a matrix can be element-wise operated on using any one of the basic arithmetic operations.

MATLAB

Code: linalg_matrices.m

Lines 127 to 155

Keywords: reshape, repmat, bsxfun

```
%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Introduction to matrices
% VIDEO: Broadcasting matrix arithmetic
% Instructor: sincxpress.com
%
%%

% create a matrix
A = reshape(1:12,3,4);

% and two vectors
```

```

r = [ 10 20 30 40 ];
c = [ 100 200 300 ]';

%% three methods for broadcasting

% the repmat way
A + repmat(r,size(A,1), 1)
A + repmat(c,1,size(A,2))

% the bsxfun way (bsxfun stands for binary expansion)
bsxfun(@plus,A,r)
bsxfun(@plus,A,c)

% the non-mathty way
A + r
A + c

%% end.

```

SECTION 5: MATRIX MULTIPLICATIONS

See included exercises with code download:

- linalg_matrixMult.pdf
- linalg_matrixMult.m
- linalg_matrixMult.ipynb

44. INTRODUCTION TO STANDARD MATRIX MULTIPLICATION

One of the basic operations in linear algebra is matrix multiplication $C = AB$, computing the product of an $m \times n$ matrix A with an $n \times p$ matrix B to produce an $m \times p$ matrix C .

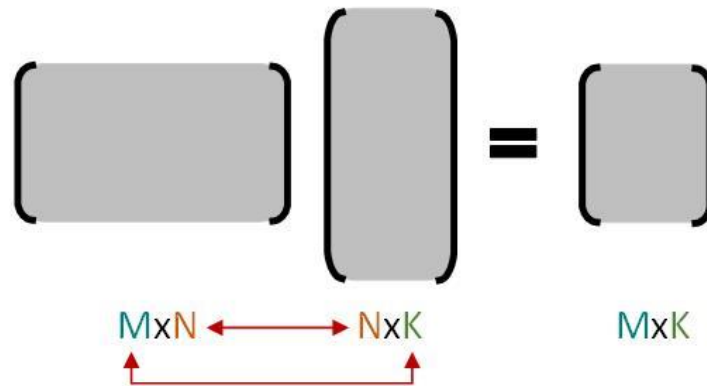
Order is important. Matrix multiplication is not commutative: $A \times B \neq B \times A$

AB can be stated as:

- A left-multiplies B
- A pre-multiplies B
- B right-multiplies A
- B post-multiplies A

Matrix multiplication is valid only when the inner dimensions match. Resulting product will have the size of the outer dimensions.

Standard matrix multiplication: The rules for validity



BiophysicsLab.com

Figure 37 - Standard matrix multiplication: The rules for validity

MATLAB

Code: linalg_matrixMult.m

lines 1 to 31

```
%%  
% COURSE: Linear algebra: theory and implementation  
% SECTION: Matrix multiplication  
% VIDEO: Standard matrix multiplication, parts 1 & 2  
% Instructor: sincxpress.com  
%  
%%  
  
%% rules for multiplication validity  
  
m = 4;  
n = 3;  
k = 6;  
  
% make some matrices  
A = randn(m,n);  
B = randn(n,k);  
C = randn(m,k);  
  
% test which multiplications are valid.  
% Think of your answer first, then test.  
A*B; % valid  
A*A; % not valid  
A'*C; % valid  
B*B'; % valid  
B'*B; % valid
```

B^*C ; % not valid C^*B ; % not valid C'^*B ; % not valid C^*B' ; % valid
--

45. FOUR WAYS TO THINK ABOUT MATRIX MULTIPLICATION

Element perspective: Build up the product matrix one element at a time. Each element in the result matrix is computed using one row from the left matrix and one column from the right matrix. The first row and first column element in the result matrix is computed by taking the dot product of the first row of the left matrix and the first column of the second matrix. Then the first row in product matrix and second column element in the result matrix is computed by taking the dot product of the second row of the left matrix and the first column of the right matrix. Etc...

Element perspective

Take dot product:

- Rows from left matrix
- Columns from right matrix

$$\begin{pmatrix} \star & \star \\ \star & \star \\ \star & \star \end{pmatrix} \begin{pmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{pmatrix} = \begin{pmatrix} \star & \blacksquare + \star & \blacksquare \\ & & \\ & & \end{pmatrix}$$

BiophysicsLab.com

Figure 38 - Matrix multiplication - element perspective

Layer perspective: The left matrix comprises columns while the right matrix comprises rows. Then compute the outer products resulting in two intermediate matrices. These intermediate matrices are rank 1 because there is only a scalar difference between each column. Create the product matrix by summing the two intermediate matrices. This method combining rank 1 layers is closely related to the spectral theorem of matrices. This method is also the basis for the singular value decomposition (SVD).

Layer perspective

Take outer product:

- Columns from left matrix
- Rows from right matrix

Sum the two intermediate matrices

$$\begin{pmatrix} \star & \star \\ \star & \star \\ \star & \star \end{pmatrix} \begin{pmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{pmatrix} = \begin{pmatrix} \star \circ & \star \blacksquare \\ \star \circ & \star \blacksquare \\ \star \circ & \star \blacksquare \end{pmatrix} + \begin{pmatrix} & \\ & \\ & \end{pmatrix}$$
$$= \begin{pmatrix} & \\ & \\ & \end{pmatrix}$$

BiophysicsLab.com

Figure 39 - Matrix multiplication - layer perspective

Column perspective: The product matrix as a linear weighted combination of the columns of the left matrix with the weights from the right matrix. So, the first column of the product matrix is the sum of the columns from the left matrix weighted by the first row's elements from the right matrix. This interpretation is useful in statistics for linear least square model fitting. Where the left matrix is called a design matrix (simplified model of the data a.k.a. regressors), and the right matrix or possible vector will contain coefficients encoding the importance of each regressor in the left matrix. The goal of model fitting is to find the best coefficients such that the weighted combination of the columns in the left matrix with the elements in the right matrix best matches the data.

Column perspective

Linear weighted combination of the columns of left matrix with the weights from the right matrix.

$$\begin{pmatrix} \star & \star \\ \star & \star \\ \star & \star \end{pmatrix} \begin{pmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{pmatrix} = \left(\begin{pmatrix} \star \\ \star \\ \star \end{pmatrix} \circ + \begin{pmatrix} \star \\ \star \\ \star \end{pmatrix} \blacksquare \right)$$

BiophysicsLab.com

Figure 40 - Matrix multiplication - column perspective

Row perspective: Build up the product matrix one row at a time. The first row in the product matrix is the combination of the sum of rows from the right matrix with the first row from the left matrix. The n^{th} row of the product matrix is the combination of the sum of rows from the right matrix with the n^{th} row from the left matrix.

Row perspective

The n^{th} row of the product matrix is the combination of the sum of rows from the right matrix with the n^{th} row from the left matrix.

$$\begin{pmatrix} \star & \star \\ \star & \star \\ \star & \star \end{pmatrix} \begin{pmatrix} \bullet & \blacksquare \\ \bullet & \blacksquare \end{pmatrix} = \begin{pmatrix} \star \begin{bmatrix} \bullet & \blacksquare \end{bmatrix} + \star \begin{bmatrix} \bullet & \blacksquare \end{bmatrix} \\ \star \begin{bmatrix} \bullet & \blacksquare \end{bmatrix} + \star \begin{bmatrix} \bullet & \blacksquare \end{bmatrix} \\ \star \begin{bmatrix} \bullet & \blacksquare \end{bmatrix} + \star \begin{bmatrix} \bullet & \blacksquare \end{bmatrix} \end{pmatrix}$$

BiophysicsLab.com

Figure 41 - Matrix multiplication - row perspective

Four ways to think about matrix multiplication

$$\begin{aligned} \begin{pmatrix} \star & \star \\ \star & \star \\ \star & \star \end{pmatrix} \begin{pmatrix} \bullet & \blacksquare \\ \bullet & \blacksquare \end{pmatrix} &= \text{Element perspective} \quad \dots \quad = \begin{pmatrix} \star \bullet + \star \bullet & \star \blacksquare + \star \blacksquare \\ \star \bullet + \star \bullet & \star \blacksquare + \star \blacksquare \\ \star \bullet + \star \bullet & \star \blacksquare + \star \blacksquare \end{pmatrix} \\ \dots &= \text{Layer perspective} \quad = \begin{pmatrix} \star \bullet & \star \blacksquare \\ \star \bullet & \star \blacksquare \\ \star \bullet & \star \blacksquare \end{pmatrix} + \begin{pmatrix} \star \bullet & \star \blacksquare \\ \star \bullet & \star \blacksquare \\ \star \bullet & \star \blacksquare \end{pmatrix} = \dots \\ \dots &= \text{Column perspective} \quad = \begin{pmatrix} \bullet \begin{pmatrix} \star \\ \star \\ \star \end{pmatrix} + \bullet \begin{pmatrix} \star \\ \star \\ \star \end{pmatrix} & \blacksquare \begin{pmatrix} \star \\ \star \\ \star \end{pmatrix} + \blacksquare \begin{pmatrix} \star \\ \star \\ \star \end{pmatrix} \end{pmatrix} = \dots \\ \dots &= \text{Row perspective} \quad = \begin{pmatrix} \star \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} & \star \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} \\ \star \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} & \star \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} \\ \star \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} & \star \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} \end{pmatrix} = \dots \end{aligned}$$

BiophysicsLab.com

Figure 42 - Four ways to think about matrix multiplication

46. CODE CHALLENGE: MATRIX MULTIPLICATION BY LAYERING

Verify that matrix multiplication by layers gives the same result as MATLAB's matrix multiplication.

MATLAB

Code: Lesson 45 Code Challenge Mat Mult by Layering.m

```
% build the product matrix layer-wise (for-loop)
c1 = zeros(m);
for i=1:n
    % sum outer products one layer at a time
    c1 = c1 + A(:,i)*B(i,:);
end
```

47. MATRIX MULTIPLICATION WITH A DIAGONAL MATRIX

When the diagonal matrix is right multiplied (post-multiplied) by a dense matrix as in top example of figure below, resultant matrix is weighted by columns of diagonal matrix.

When the diagonal matrix is left multiplied (pre-multiplied) by a dense matrix as in the bottom example of figure below, resultant matrix is weighted by rows of diagonal matrix.

Matrix multiplication with a diagonal matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} = \begin{pmatrix} a1 & b2 & c3 \\ a4 & b5 & c6 \\ a7 & b8 & c9 \end{pmatrix}$$
$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} a1 & a2 & a3 \\ b4 & b5 & b6 \\ c7 & c8 & c9 \end{pmatrix}$$

BiophysicsLab.com

Figure 43 - Matrix multiplication with a diagonal matrix

48. ORDER-OF-OPERATIONS ON MATRICES

Given four matrices valid to multiply together (square matrices) L, I, V, E:

If you want to multiply these matrices together than take the transpose of the result,
This is the same as taking the transpose of each matrix and multiplying them together in reverse order.

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

This rule applies to any operator, not just transpose.

For some operators, not transpose, there are times when the operator creates an individual matrix than cannot be multiplied.

MATLAB

Code: linalg_matrixMult.m

lines 32 to 54

```
% result of "forward" multiplication and then transpose
res1 = (L*I*V*E)';

% result of "flipped" multiplication of transposed matrices
res2 = E'*V'*I'*L';

% test equality by subtracting (ignore possible computer rounding errors)
res1-res2
```

49. MATRIX-VECTOR MULTIPLICATION

Just like normal matrix multiplication only the vector has only one dimension.

The product of a matrix and vector will always be a vector.

Symmetric matrices are used in machine learning, statistics and multi-variate data analysis.

Matrix-vector multiplication: always a vector

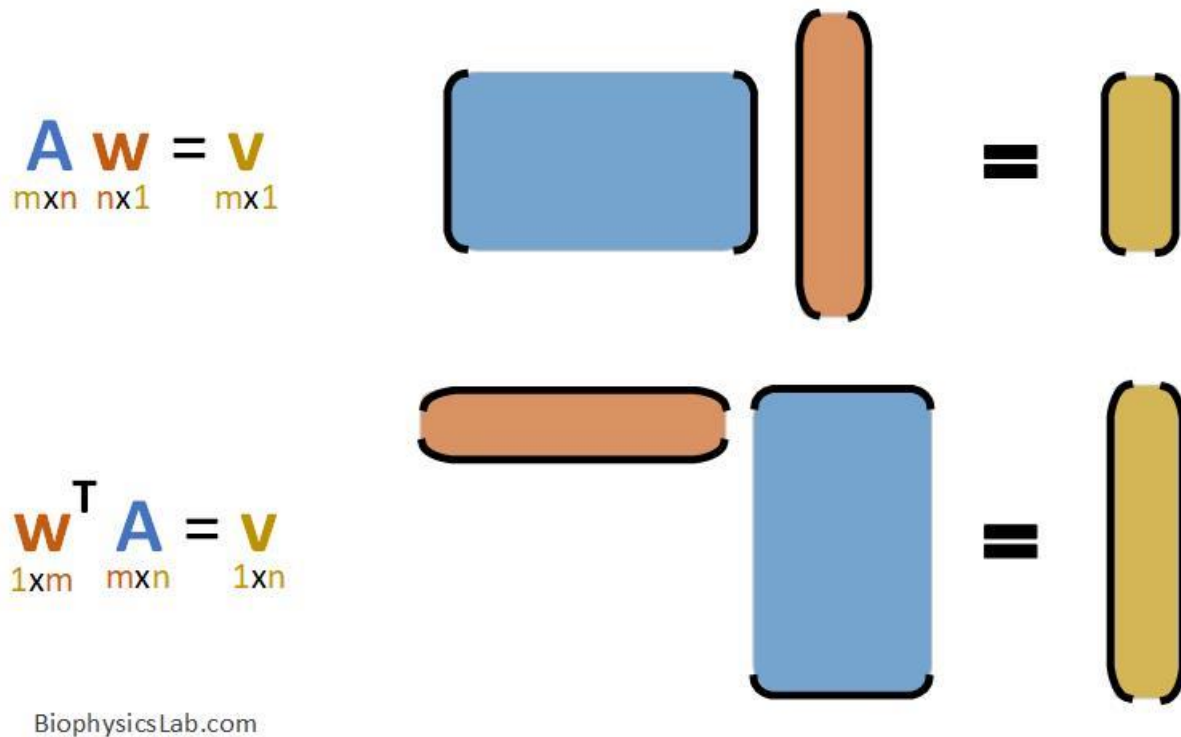


Figure 44 - Matrix-vector multiplication: always a vector

Some key points about matrix vector multiplication:

- Matrix vector multiplication gives different results depending on whether the vector is pre-multiplied (on left of matrix) or post-multiplied (on right of matrix).
- When right multiplying a vector, the result gives weighted combinations of the columns of the matrix. See figure below.
- When pre-multiplying by the vector, the result gives weighted combinations of the rows of the matrix. Again, see figure below.

Matrix-vector multiplication example

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} a2 + b3 \\ c2 + d3 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a2 + c3 \\ b2 + d3 \end{bmatrix}^T$$

BiophysicsLab.com

Figure 45 - Matrix-vector multiplication example

Matrix-vector multiplication with symmetry

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} a2 + b3 \\ b2 + c3 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} a2 + b3 \\ b2 + c3 \end{bmatrix}^T$$

BiophysicsLab.com

Figure 46 - Matrix-vector multiplication with symmetry

MATLAB

Code: linalg_matrixMult.m

lines 55 to 86

```
% number of elements
m = 4;

% create matrices
N = round( 10*randn(m) );
S = round( N'*N/m^2 ); % scaled symmetric

% and vector
w = [-1 0 1 2]'; % transpose to get a column vector

% with symmetric matrix
S*w % 1
(S*w)' % 2
% w*S % 3 Invalid matrix sizes
w'*S' % 4
w'*S % 5

% with nonsymmetric matrix
N*w % 1
(N*w)' % 2
% w*N % 3 Invalid matrix sizes
w'*N' % 4
w'*N % 5
```

50. 2D TRANSFORMATION MATRICES

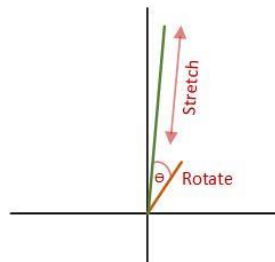
Describe matrix vector multiplication application using geometry.

Demonstrate multiplication leading to both rotation and stretch.

2D transformation matrices: rotation and stretch

$$\begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 11 \end{pmatrix}$$

Machine Input data Output data



BiophysicsLab.com

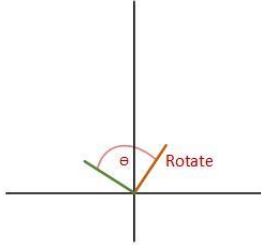
Figure 47 - 2D transformation matrices: rotation and stretch

Demonstrate multiplication leading to just rotation.

2D transformation matrices: pure rotation

$$\begin{pmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{pmatrix} \xrightarrow{\Theta = \pi/2} \begin{pmatrix} \cos(\pi/2) & -\sin(\pi/2) \\ \sin(\pi/2) & \cos(\pi/2) \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \end{pmatrix}$$

Rotation matrix Machine Input data Output data



BiophysicsLab.com

Figure 48 - 2D transformation matrices: rotation

Demonstrate multiplication leading to only stretch – which is also an important decomposition technique called an eigenvector and eigenvalue.

2D transformation matrices: stretch

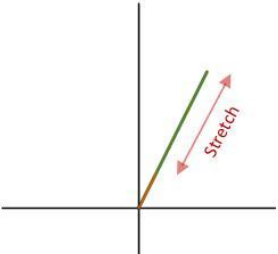
$$\begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \\ 8 \end{pmatrix}$$

Machine Input data Output data

$$4 \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \\ 8 \end{pmatrix}$$

Eigenvalue Eigenvector

λ \mathbf{v}



$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$

Fundamental
Eigenvalue equation

BiophysicsLab.com

Figure 49 - 2D transformation matrices: stretch

Think of the matrix as a “machine” and the vector to multiply as data input. The function of the machine is to apply a transformation. The resulting vector is the output from the machine. Geometrically the output is some combination of rotating and stretching.

MATLAB

Code: linalg_matrixMult.m
lines 87 to 158

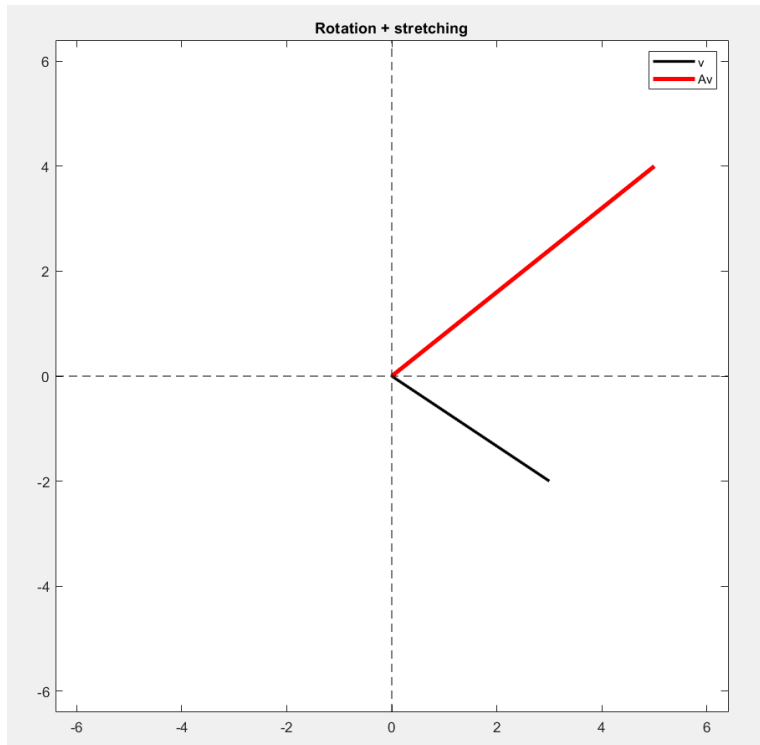


Figure 50 - MATLAB 2D rotation and stretch transformation matrix Av : $v = [3 \ -2]$; $A = [1 \ -1; 2 \ 1]$

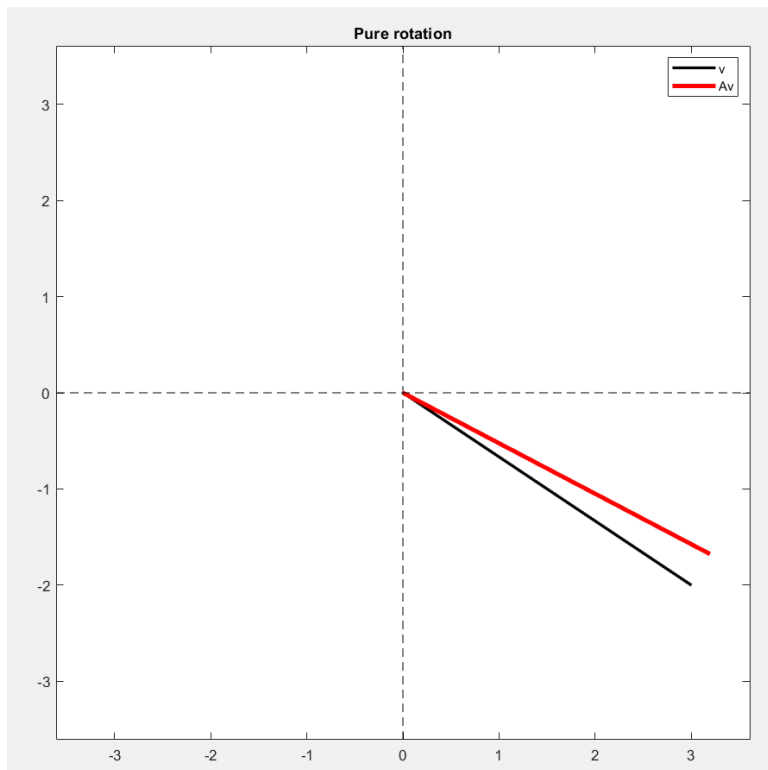


Figure 51 - MATLAB 2D rotation transformation matrix Av : $v = [3 \ -2]$; $A = [\cos(\theta) \ -\sin(\theta); \sin(\theta) \ \cos(\theta)]$

51. CODE CHALLENGE: PURE AND IMPURE ROTATION MATRICES

MATLAB

Code: Lesson_50_Code_Challenge_Rotation.m

Keywords: linspace, zeros, length, cos, sin, clf, plot, ylabel, xlabel, legend

```
%% Explore length of pure and impure rotation matrices
```

```
% 2D input vector  
v = [ 3 -2 ]';
```

```
thetas = linspace(0,2*pi,100);
```

```
% initialize  
vecmags = zeros(length(thetas),2);
```

```
for i=1:length(thetas)
```

```
    % rotation angle (specify in radians)  
    theta = thetas(i);
```

```
    % 2x2 transformation matrix (impure rotation matrix)  
    A1 = [ 2*cos(theta) -sin(theta);  
          sin(theta) cos(theta) ];
```

```
    % 2x2 transformation matrix (pure rotation matrix)  
    A2 = [ cos(theta) -sin(theta);  
          sin(theta) cos(theta) ];
```

```
    % output vector is Av  
    vecmags(i,1) = norm( A1*v );  
    vecmags(i,2) = norm( A2*v );  
end
```

```
% plotting  
clf  
plot(thetas,vecmags,'linew',3)  
ylabel('Av magniture'), xlabel('Angle (rad.)')  
legend({'impure rotation';'pure rotations'})
```

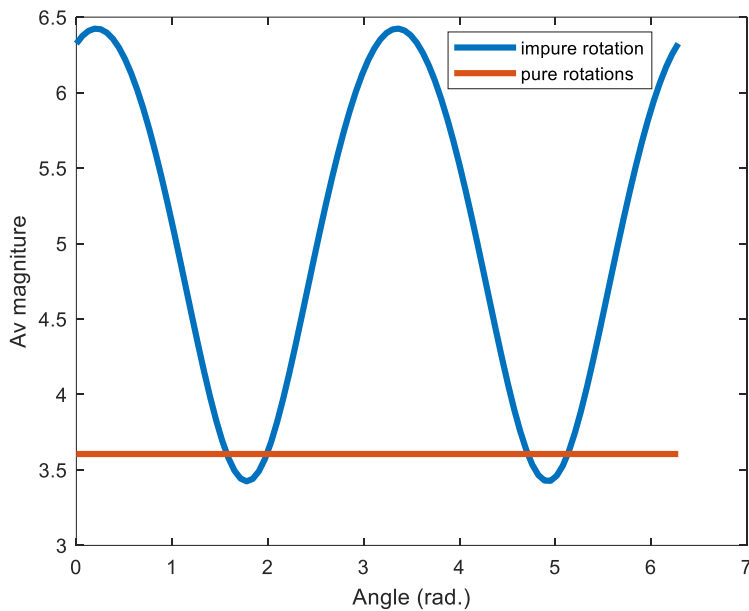


Figure 52 - MATLAB Code challenge: Pure and impure rotation matrices

52. CODE CHALLENGE: GEOMETRIC TRANSFORMATIONS VIA MATRIX MULTIPLICATIONS

Matrix multiplication for geometric transformations starting with a circular coordinate system $[\cos(x); \sin(x)]$: identity, some test matrices, singular, and zero:

- The geometric implications of a singular matrix: A singular matrix has columns that form a rank 1 linearly dependent set.
- The geometric implications of an identity matrix: An identity matrix will not change the vector multiplication.
- The geometric implications of a zero matrix: A zero matrix will generate only one rank 0 data point.

MATLAB

Code: Lesson_51_Code_Challenge_Geometric_Transformations.m

Keywords: figure, clf, hold on, plot, 'markerfacecolor', axis, axis square, legend

```
%% Code Challenge: Geometric transformations via multiplications
```

```
% Generate x,y coordinates for a circle and Plot
```

```
x = linspace(-pi,pi,100);
```

```
xy = [cos(x); sin(x)]';
```

```
figure(1)
```

```
clf
```

```
plot(xy(:,1),xy(:,2),'bs','markerfacecolor','w')
```

```
% Create a 2x2 matrix identity matrix, I
```

```

% Note: I matrix produces the same circle plot as above
I = [1 0; 0 1];

% Multiply matrix by same x,y coordinates and Plot
Identxy = xy*I;
hold on
plot(Identxy(:,1),Identxy(:,2),'r','markerfacecolor','m')

% Create a new matrix, multiply by same x,y coordinates and Plot
Test1 = [1 0; 1 2];
Test1xy = xy*Test1;
hold on
plot(Test1xy(:,1),Test1xy(:,2),'rs','markerfacecolor','m')

% Try with another matrix
hold on
Test2 = [1 0; 0 .4];
Test2xy = xy*Test2;
hold on
plot(Test2xy(:,1),Test2xy(:,2),'b*')

% Try with a singular matrix (columns form a rank 1 linearly dependent set)
S = [1 2; 2 4];
Sxy = xy*S;
hold on
plot(Sxy(:,1),Sxy(:,2),'bo')

% Try with a rank zero matrix
Z = [0 0; 0 0];
Zxy = xy*Z;
hold on
plot(Zxy(:,1),Zxy(:,2),'gs','markerfacecolor','g')

% Use a suitable axis with legend
axis([-1 1 -1 1]*max(abs([ xy(:); Sxy(:) ])))
axis square
legend('xy: [cos(x); sin(x)]', 'Identity: [1 0; 0 1]*xy', 'Test1: [1 0; 1 2]*xy', 'Test2: [1 0; 0 .4]*xy', 'Singular: [1 2; 2 4]*xy', 'Zero: [0 0; 0 0]*xy')

```

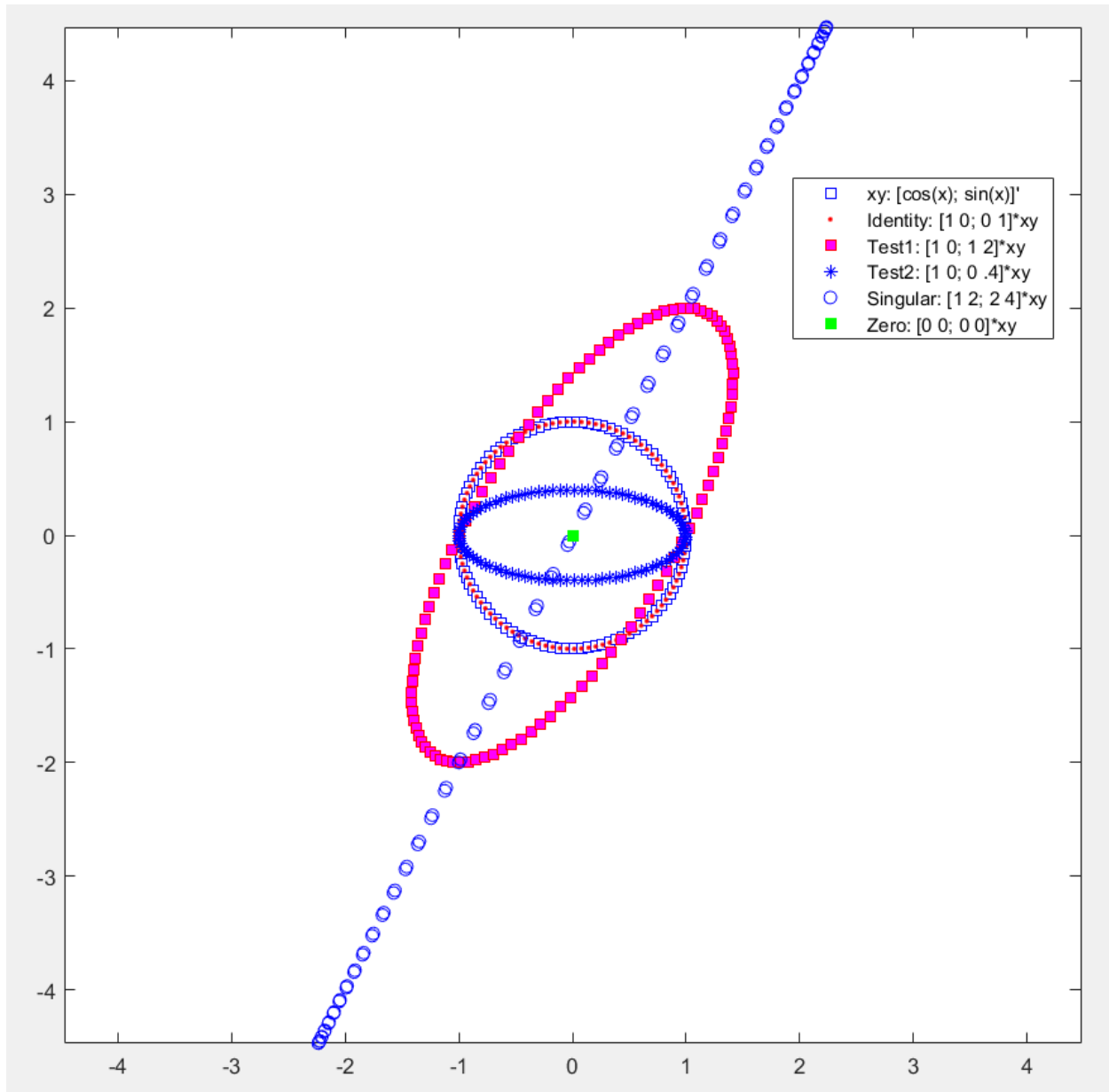


Figure 53 - MATLAB Code challenge: Geometric transformations via matrix multiplications

53. ADDITIVE AND MULTIPLICATIVE MATRIX IDENTITIES

Multiplicative identity matrix: I matrix

$$\mathbf{A}\mathbf{I} = \mathbf{I}\mathbf{A} = \mathbf{A}$$

$$\mathbf{A} + \mathbf{I} \neq \mathbf{A}$$

Additive identity matrix: zeros matrix

$$\mathbf{A}\mathbf{0} = \mathbf{0}\mathbf{A} \neq \mathbf{A}$$

$$\mathbf{A} + \mathbf{0} = \mathbf{A}$$

MATLAB

Code: linalg_matrixMult.m

lines 159 to 183

Keywords: round, randn, eye, zeros, isequal

```
% test both identities (answer is 1=true or 0=false)
isequal(A*I, A) % True
isequal(A, A*I) % True
isequal(A, A+I) % False

isequal(A, A+Z) % True
isequal(A+Z, A*I) % True
```

54. ADDITIVE AND MULTIPLICATIVE SYMMETRIC MATRICES

A symmetric matrix is a square matrix that is equal to its transpose.

Symmetric matrices are commonly used in statistics, data analysis and machine learning.

Symmetric matrices via the additive method:

$$\mathbf{S} = (\mathbf{A} + \mathbf{A}^T)/2$$

$$\mathbf{S} = \mathbf{S}^T$$

Where \mathbf{A} is a non-symmetric square matrix and /2 is an optional scaling method.

And \mathbf{S} is a symmetric square matrix

Symmetric matrix via the additive method: example

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} + \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix} = \begin{pmatrix} a+a & b+d & c+g \\ b+d & e+e & f+h \\ c+g & h+f & i+i \end{pmatrix}$$

Non-symmetric square
Non-symmetric square transposed
Symmetric square

BiophysicsLab.com

Figure 54 - Symmetric matrix via the additive method: example

Multiplicative symmetric matrices ($\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$). Great application value. It is basis for the covariance matrix in signal processing and statistics. Also used in principle components analysis (PCA), and linear least square modeling. Assume columns are time and rows are sensors for example. The resulting matrix is called a covariance matrix where the diagonal elements contain the variance of each channel (row) and the off-diagonal elements contain each of the covariances between each of the channels.

$$\mathbf{S}_1 = \mathbf{A}^T \mathbf{A}$$

$$\mathbf{S}_2 = \mathbf{A} \mathbf{A}^T$$

Where \mathbf{A} is a non-symmetric square or rectangular matrix.
 And \mathbf{S}_1 and \mathbf{S}_2 are symmetric square matrices.
 \mathbf{S}_1 and \mathbf{S}_2 are equal only if \mathbf{A} is a square matrix.

Multiplicative symmetric matrix method: example

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} a & d \\ b & e \\ c & f \end{pmatrix} = \begin{pmatrix} a^2 + b^2 + c^2 & ad + be + cf \\ ad + be + cf & d^2 + e^2 + f^2 \end{pmatrix}$$

Non-symmetric rectangle Non-symmetric rectangle transposed Symmetric square

BiophysicsLab.com

Figure 55 - Multiplicative symmetric matrix method: example

MATLAB

Code: linalg_matrixMult.m

lines 184 to 224

Keywords: size

Demonstrate additive and multiplicative methods in MATLAB code.

55. HADAMARD (ELEMENT-WISE) MULTIPLICATION

Two matrices must be the same size. Element-wise multiplication is sometimes used in probability theory – such as in computing the joint probability between two events. It is also used as a convenient notation when a lot of individual multiplications are needed.

Hadamard (element-wise) multiplication

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 6 & 3 \end{pmatrix} \odot \begin{pmatrix} 3 & 8 & 5 \\ 4 & 1 & -5 \end{pmatrix} = \begin{pmatrix} 0 & 8 & 10 \\ -4 & 6 & 15 \end{pmatrix}$$

BiophysicsLab.com

Figure 56 - Hadamard (element-wise) multiplication

Note: See 19. Vector Hadamard multiplication

MATLAB

Code: `linalg_matrixMult.m`
lines 225 to 244

```
% any matrix sizes
m = 13;
n = 2;

% but the two matrices must be the same size
A = randn(m,n);
B = randn(m,n);

% note the .* instead of *
C = A .* B
```

56. CODE CHALLENGE: SYMMETRY OF COMBINED SYMMETRIC MATRICES

MATLAB

Code: `Lesson 55 Code Challenge Symmetry.m`

```
%% 55. Code challenge: symmetry of combined symmetric matrices

% Create two symmetric matrices
m = 3;
A = randn(m);
B = randn(m);

% Make square (if not already) and symmetric
AtA = A'*A;
BtB = B'*B;

% Compute sum, multiplication, and Hadamard multiplication of the two
```

```
% matrices
Cs = AtA + BtB;
Cm = AtA * BtB;
Ch = AtA .* BtB;

% Determine whether the result is still symmetric
Cs - Cs' % Yes (Because operations are element-wise)
Cm - Cm' % No (Because operations are row-wise and column-wise)
Ch - Ch' % Yes
```

57. MULTIPLICATION OF TWO SYMMETRIC MATRICES

Normally two square symmetric matrices multiplied together do not produce a symmetric resultant matrix.

But if the two matrices are only 2x2 and the diagonal elements are equal within each symmetric matrix then the resultant matrix will also be symmetric. This constraint with equal diagonal elements does not work for 3x3 and larger matrices.

Uses: Multivariate decoding, Fisher discriminant decoding analysis (Fisher linear discriminant, FLD; Fisher 1936).

Symmetric matrices produce asymmetric result

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0a+1c & 0b+1d \\ 2a+3c & 2b+3d \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0a+1c & 0b+1d \\ 2a+3c & 2b+3d \end{pmatrix}$$

BiophysicsLab.com

Figure 57 - Symmetric matrices produce asymmetric result

MATLAB

Code: linalg_matrixMult.m
lines 245 to 283

Keywords: syms, diag

Note: Requires MATLAB Symbolic Toolbox, Python can use sympy library

```
syms a b c d e f g h i j k l m n o p q r s t u

% symmetric and constant-diagonal matrices
A = [ a b c d;
      b a e f;
      c e a h;
      d f h a ];

B = [ l m n o;
      m l q r;
      n q l t;
      o r t l ];

% confirmation that A and B are symmetric
A - A.'
B - B.'

% ... and constant diagonal
diag(A)
diag(B)

% but AB neq (AB)'
A*B - (A*B).'

% maybe for a submatrix?
n = 3;
A1 = A(1:n,1:n);
B1 = B(1:n,1:n);

A1*B1 - (A1*B1).'
```

58. CODE CHALLENGE: STANDARD AND HADAMARD MULTIPLICATION FOR DIAGONAL MATRICES

Applications for diagonal matrix multiplication: Eigenvalue and Singular Value Decomposition
MATLAB

Code: Lesson_57_Code_Challenge_Diagonal_Mult.m

Keywords: randn, diag

```
%% 57. Code challenge: standard and Hadamard multiplication for diagonal matrices

% Create two matrices (4x4): "full" and diagonal
A = randn(4);
D = diag( rand(4,1) ); % Input vector becomes the elements of the diagonal

% Multiply each matrix by itself (A*A): standard and Hadamard
% These two multiplications produce different results
A*A % standard
A.*A % element-wise Hadamard
```

```
% These two multiplications produce the same result
D*D
D.*D
```

59. CODE CHALLENGE: FOURIER TRANSFORM VIA MATRIX MULTIPLICATION

Fourier transform:

$$\mathbf{F}_{j,k} = \omega^m$$

$$\omega = e^{-2\pi\sqrt{-1}/n}$$

$$m = (j - 1)(k - 1)$$

$$\mathbf{X} = \mathbf{F}\mathbf{x}$$

Where:

\mathbf{F} is the Fourier matrix, a complex square matrix with rank n

\mathbf{x} is a vector

\mathbf{X} is the vector of Fourier coefficients

MATLAB

Code: Lesson_58_Code_Challenge_Fourier_Transform.m

Keywords: zeros, exp, i, randn, fft, imagesc, real, imag, plot

```
n = 52;
F = zeros(n);

w = exp(-2*pi*1i/n);

for j=1:n
    for k=1:n
        m = (j-1)*(k-1);
        F(j,k) = w^m;
    end
end

x = randn(n,1);
x1 = F*x;
x2 = fft(x);
```

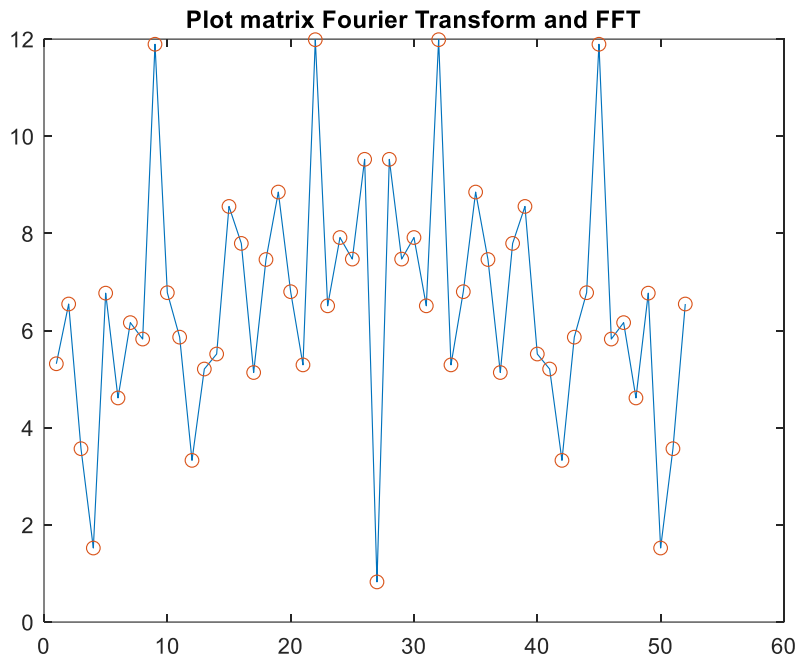


Figure 58 - MATLAB Fourier Transform results

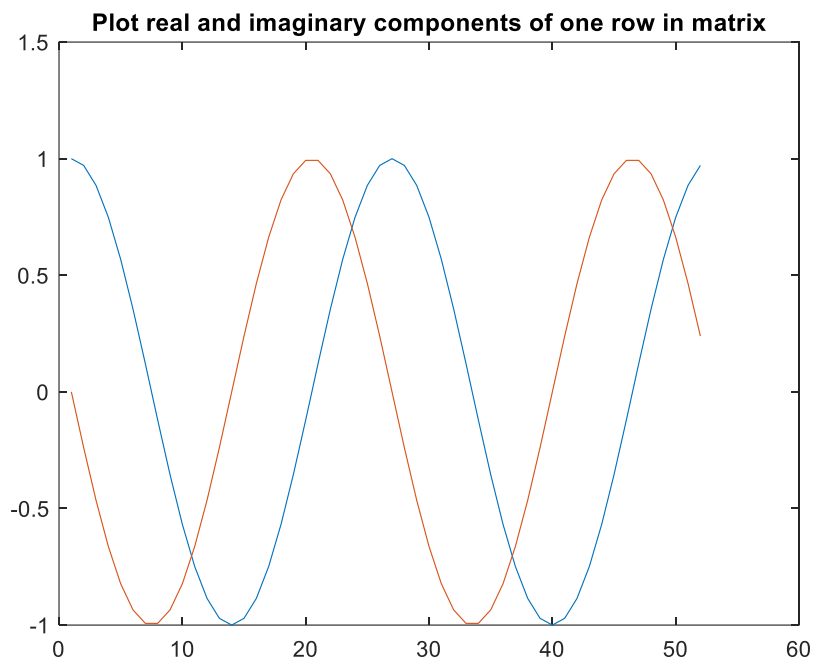


Figure 59 - MATLAB Sine (imaginary) and Cosine (real) results

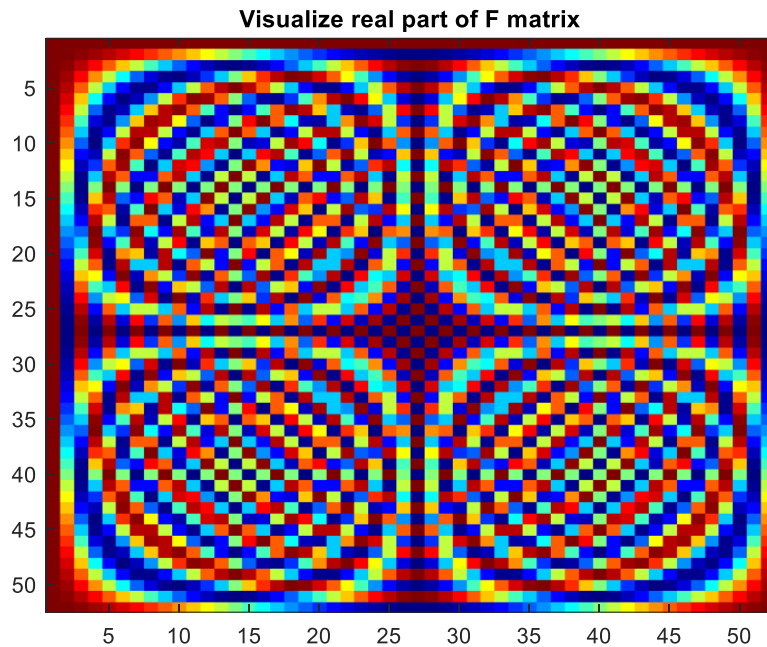


Figure 60 - MATLAB view real components of matrix using imagesc

60. FROBENIUS DOT PRODUCT

Defined for two matrices with the same dimensions.

Three ways to compute the Frobenius dot product.

Option 1

Step 1: Element-wise multiplication (of entire matrices).

Step 2: Sum all elements.

Option 2:

Step 1: Vectorize both matrices.

Step 2: Compute vector dot product.

Option 3 (most common method):

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \text{tr}(\mathbf{A}^T \mathbf{B})$$

Frobenius norm, a.k.a. Euclidean norm:

$$\text{norm}(\mathbf{A}) = \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle_F} = \sqrt{\text{tr}(\mathbf{A}^T \mathbf{A})}$$

MATLAB

Code: linalg_matrixMult.m

lines 284 to 320

Keywords: trace, norm, sqrt

```
% Option 1 (in notes) brute force method
sumAB = 0;
for i=1:m
```

```

    for j=1:n
        sumAB = sumAB + A(i,j)*B(i,j);
    end
end
frob_dp0 = sumAB;

% Option 2 (in notes) first vectorize, then vector-dot-product
Av = A(:);
Bv = B(:);
frob_dp = sum( Av.*Bv );

% Option 3 (in notes) trace method
frob_dp2 = trace( A'*B );

% matrix norm
Anorm = norm(A,'fro');
Anorm2 = sqrt( trace( A'*A ) );

```

61. MATRIX NORMS

There are many matrix norms that give different answers.

Only one vector norm.

Three families of matrix norms

1. Frobenius norm, a.k.a. Euclidean norm (see [lecture 60](#))

$$\text{norm}(\mathbf{A}) = \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle_F} = \sqrt{\text{tr}(\mathbf{A}^T \mathbf{A})}$$

2. Induced 2-norm

A measure of how much matrix \mathbf{A} scales vector \mathbf{x} .

$$\|\mathbf{A}\|_p = \sup \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_p}, \quad \mathbf{x} \neq 0$$

Where:

\sup is the supremum (like maximum in most cases)
 $p=2$ for the 2-norm

3. Schatten p-norm

$$\|\mathbf{A}\|_p = \left(\sum_{i=1}^r \sigma_i^p \right)^{1/p}$$

Where:

σ = singular values of a matrix

Reference: A top nine list: Most popular induced matrix norms, Andrew D. Lewis 2010/03/20

MATLAB

Code: linalg_matrixMult.m

lines 321 to 351

Keywords: qr, randn, norm, svd, sum, disp

```
% VIDEO: Matrix norms
% Instructor: sincxpress.com
%

% Create a matrix
A = [ 1 2 3; 4 5 6; 7 7 9 ];
% optional orthogonal matrix to show that 2-norm is 1
% Note: an orthogonal matrix means every column in a matrix is orthogonal
% to every other column, and the magnitude (norm) of each column is 1.
% [Q,R] = qr(randn(5));
% A = Q;

% Frobenius norm
normFrob = norm(A,'fro');

% induced 2-norm
normInd2 = norm(A);
% note: computed as below
% lambda = sqrt( max(eig(A'*A)) );

% Schatten p-norm
p = 1;
s = svd(A); % get singular values
normSchat = sum(s.^p)^(1/p);

% show all norms for comparison
disp([ normFrob normInd2 normSchat ])
```

Output from disp showing that each norm method produces different results:

16.4317 16.3920 17.8182

62. CODE CHALLENGE: CONDITIONS FOR SELF-ADJOINT

Self-adjoint properties appears in multi-variate calculus, differential geometry, physics, and other computations on a curved surface.

$$\langle \mathbf{A}\mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{v}, \mathbf{A}\mathbf{w} \rangle \quad \mathbf{v} \neq \mathbf{w}$$

Where $\langle \text{angle brackets} \rangle$ indicate dot product.

Code should:

1. List 2-3 conditions for this equality to hold.
2. Prove the equality when those conditions are met.
3. Illustrate in code.

Three conditions:

1. A is square, $m \times m$.
2. A is symmetric.
3. v and w are the same size ($m \times 1$).

MATLAB

Code: Lesson_61_Code_Challenge_Self_Adjoint.m

```
%% 61. Code challenge: conditions for self-adjoint

m = 5;

A = randn(m);
A = A*A'; % make A symmetric

v = randn(5,1);
w = randn(5,1);

dot(A*v,w) - dot(v,A*w) % equals zero
```

63. CODE CHALLENGE: THE MATRIX ASYMMETRY INDEX

Learn a measure of how symmetric a matrix is

Learn how to create a skew-symmetric (perfectly asymmetric) matrix

Steps to the code challenge:

1. Implement the matrix asymmetry index in code
2. Test on:
 - a. Symmetric matrix
 - b. Skew-symmetric matrix
 - c. Random matrix (use additive method to create random symmetric matrices)
3. Develop a formula that will proportionally mix a symmetric and skew-symmetric matrix
4. Conform that the formula works using random matrices

Matrix asymmetry index

$$a_i = \|\tilde{A}\| / \|A\| \quad \text{Ratio of norms}$$

$$\tilde{A} = (A - A^T)/2 \quad \text{Asymmetric part of } A$$

For a perfectly symmetric matrix $\xrightarrow{\text{yields}} a_i = 0$

For a skew-symmetric matrix $\xrightarrow{\text{yields}} a_i = 1$

$$\mathbf{B} = (1 - p)(\mathbf{A} + \mathbf{A}^T) + p(\mathbf{A} - \mathbf{A}^T)$$

Where:

$$1 \geq p \geq 0$$

When $p = 0$, matrix symmetric

When $p = 1$, matrix is skew-symmetric

MATLAB

Code: Lesson_63_New_Code_Challenge_Matrix_Asymmetry_Index.m

Keywords: zeros, linspace, i, norm, randn, round, length, size, plot

```
%% Part 1: implement MAI (Matrix Asymmetry Index)

A_anti = (A-A') / 2;
MAI = norm(A_anti) / norm(A);

%% Part 2: compute MAI for symmetric, skew-symmetric, and random matrix

A = randn(5);

% Additive method:
% create symmetric matrix from a non-symmetric square matrix
A = (A+A')/2;
MAI = norm((A-A')/2) / norm(A);

% create asymmetric matrix
A = (A-A')/2;
MAI = norm((A-A')/2) / norm(A);

% create random matrix
MAI = norm((A-A')/2) / norm(A);

%% Part3: formula for mixing skew/symmetric matrices

A = round(10*randn(5));
%p = 0; % 0 = symmetric,
%p = 1; % 1 = skew-symmetric,
p = .3; % between 0 & 1 = random
A = (1-p)*(A+A') + p*(A-A');

%% part 5: test on random matrices
```

```

ps = linspace(0,1,100);
mai = zeros(size(ps));

for i=1:length(ps)
    p = ps(i);
    A = randn(5);
    A = (1-p)*(A+A') + p*(A-A');
    mai(i) = norm((A-A')/2) / norm(A);
end

plot(ps,mai,'s-')

```

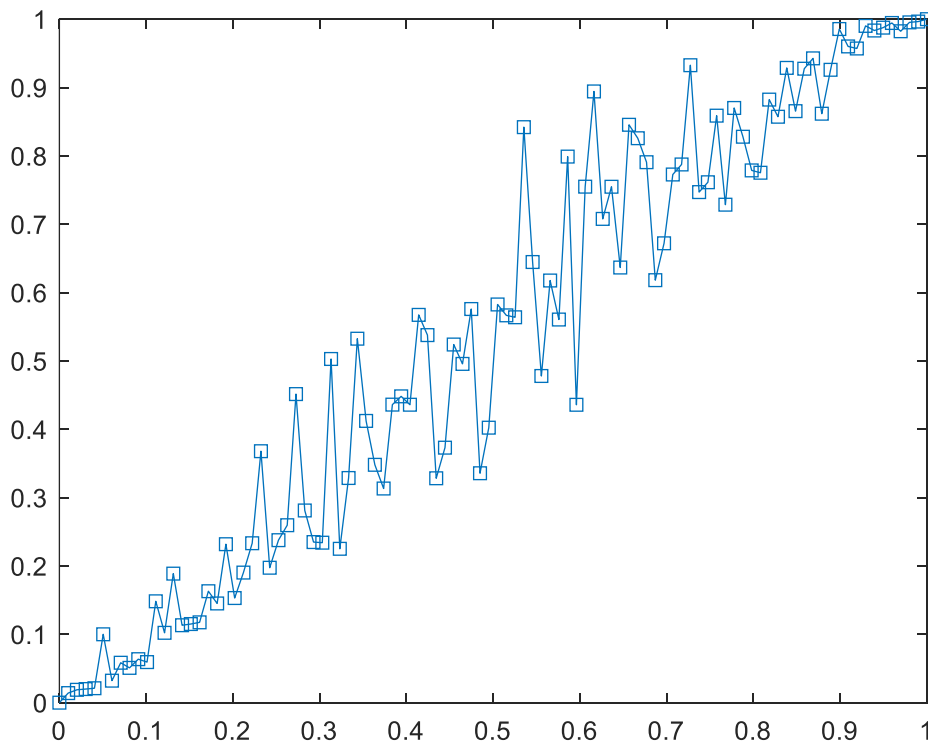


Figure 61 - Test on random matrices

64. WHAT ABOUT MATRIX DIVISION?

Element-wise division.

Used in data analysis and statistics. Ex: left matrix might have sensor signal intensity values, and right matrix might have noise estimates, then quotient matrix would be the signal to noise ratio (S/N). Yet these data collections aren't really matrices, just data collections. So, we aren't really dividing one matrix by another.

Constraints:

- Both matrices must be the same size.
- Denominator matrix (right) must not contain any zeros.

Hadamard (element-wise) division

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 6 & 3 \end{pmatrix} \oslash \begin{pmatrix} 3 & 8 & 5 \\ 4 & 1 & -5 \end{pmatrix} = \begin{pmatrix} 0 & 1/8 & 2/5 \\ -1 & 6 & -3/5 \end{pmatrix}$$

BiophysicsLab.com

Figure 62 - Hadamard (element-wise) division

Matrix division using matrix inverse.

SECTION 6: MATRIX RANK

See included exercises with code download:

- linalg_matrixRank.pdf
- linalg_matrixRank.m
- linalg_matrixRank.ipynb

66. RANK: CONCEPTS, TERMS, AND APPLICATIONS

1 Rank (related to the dimensionality of a matrix):

r or **rank(A)**

Where:

- **A** can be square or rectangular matrix,
- A non-negative integer (0, 1, 2, 3, ...).

#2 Maximum possible range:

max(r) = min(m, n)

Where:

- **m** is the number of rows,
- **n** is the number of columns.

More formally:

$$r \in \mathbb{N}, \text{ s.t. } 0 \leq r \leq \min \{m, n\}$$

#3 Rank is a property of the matrix:

Rank is not considered by column or row, just the matrix as a whole.

#4 More rank terminology:

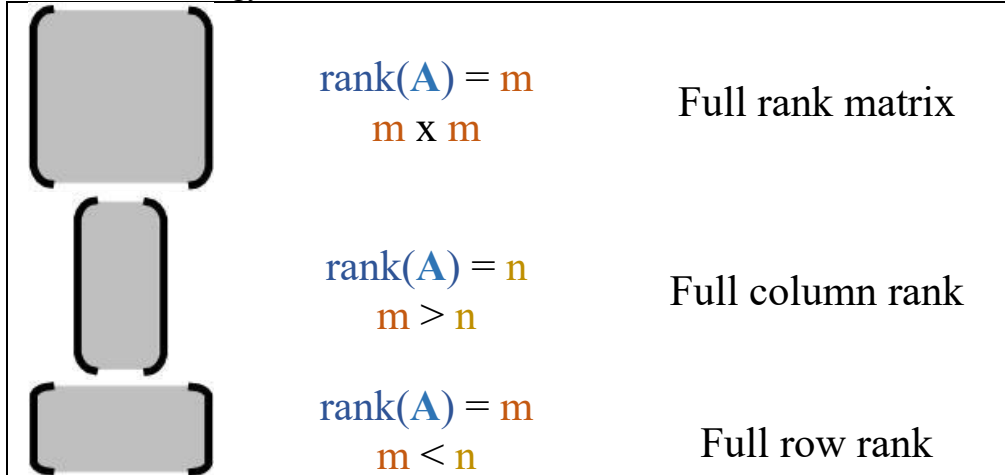


Figure 63 - Rank terminology

If the matrix rank is less than the maximum possible rank then it can be called:

- Reduced rank
- Rank deficient
- Degenerate
- Low-rank
- Singular or uninvertible (when matrix is square)

#5 Rank = dimensionality of information

Not the same thing as the ambient dimensionality or total number of rows/columns.

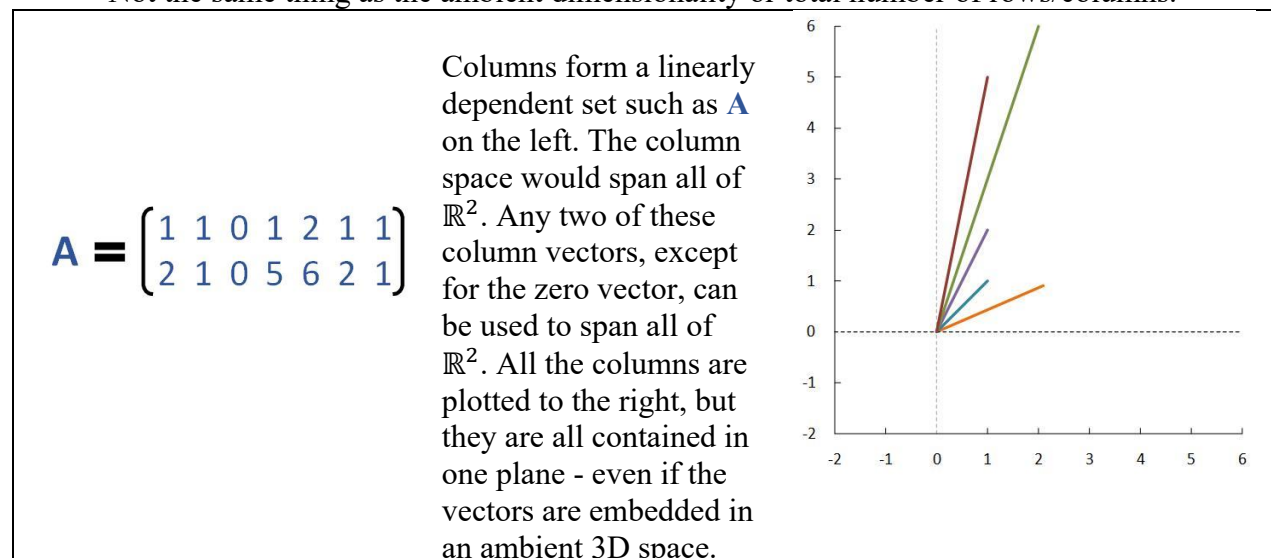


Figure 64 - Rank dimensionality of information: algebraic and geometric

#6 One definition of rank

The rank of a matrix is the largest number of columns (or rows) that can form a linearly independent set.

67. COMPUTING RANK: THEORY AND PRACTICE

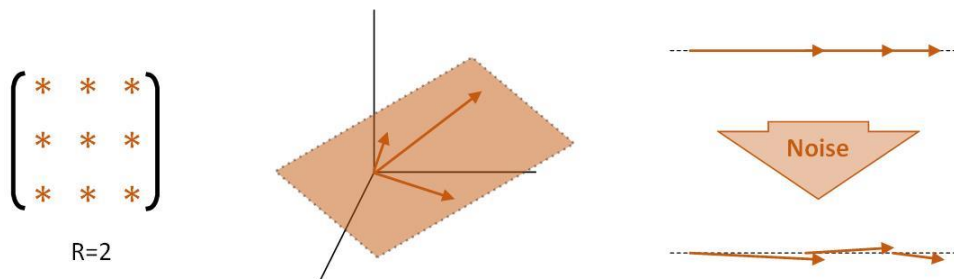
Methods to compute rank:

1. Count the number of columns (or rows) in a linearly independent set.
 - a. By visual inspection and a bit of guesswork...
 - b. By applying methods used to solve systems of simultaneous linear equations.
2. Apply row reduction to reduce matrix to echelon form (more on this later), and count the number of pivots.
 - a. Tedious and time-consuming for large matrices or matrices with decimal entries.
3. Compute the singular value decomposition (SVD) and count the number of non-zero singular values.
4. Compute the eigendecomposition and count the number of non-zero eigenvalues.

Difficulties in computing rank of matrices in practice:

1. Computer rounding errors (e.g., $0 \geq 10e-15$)
 - a. Example: 50x50 rank-46 matrix. Is $\lambda=10e-13$ real or 0?
2. Noise.

Difficulties in computing rank of matrices in practice



BiophysicsLab.com

Figure 65 - Difficulties in computing rank of matrices in practice

The figure above shows how noise affects rank calculation for a matrix. Object to the left shows an example of sensor data with some redundancy so even though there are 3 rows and 3 columns, the rank is known to be 2. The three vectors are shown to exist on a 2D plan in the center object, verifying a rank of 2. The right-hand object shows the 2D plan visualized on edge. With no noise all the vectors remain on a 1D line. But when noise is considered, there may be some Z component to the vector causing the 3 vectors to be elevated from the line of the plane. In this case the rank with noise might seem like a value of 3 when the rank is really a value of 2 when the noise is removed.

MATLAB

Code: linalg_matrixRank.m

Lines: 1 to 49

Keywords: rank, disp, num2str, randn, end, round

Output from column and row penultimate copy code:

```
rank(A) = 4
```

```
rank(B) = 4
```

```
rank(C) = 3
```

Output from noise code:

```
rank(w/o noise) = 3
```

```
rank(with noise) = 4
```

68. RANK OF ADDED AND MULTIPLIED MATRICES

Know the limits (maximum possible value) of $\text{rank}(\mathbf{A}+\mathbf{B})$ and $\text{rank}(\mathbf{AB})$.

For addition or subtraction:

$$\text{rank}(\mathbf{A}+\mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$$

For multiplication

$$\text{rank}(\mathbf{AB}) \leq \min \{ \text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}) \}$$

These rules apply to both square and rectangular matrices.

69. CODE CHALLENGE: REDUCED-RANK MATRIX VIA MULTIPLICATION

MATLAB

Code: Lesson_67_Code_Challenge_Rank_Mult.m

Keywords: randn, size, rank

```
%% 67. Code challenge: reduced-rank matrix via multiplication

% Create reduced-rank matrices using matrix multiplication

% Create a 10x10 matrix with rank=4 using matrix multiplication
% A = randn(10,4) * randn(4,10)
% size(A)
% rank(A)

% Generalize the procedure to create any MxN matrix with rank r
m = 4;
n = 4;
r = 1; % Note: r must be less than or equal to smallest matrix dimension

A = randn(m, r) * randn(r, n)
size(A)
rank(A)
```

70. CODE CHALLENGE: SCALAR MULTIPLICATION AND RANK

MATLAB

Code: Lesson 67 Code Challenge Rank Scalar.m

```
%% 68. Code challenge: scalar multiplication and rank

% Test whether the matrix rank is invariant to scalar multiplication
% Yes: except when I = 0

% Create two matrices: full-rank and a reduced-rank (random)
m = 6;
n = 4;

F = randn(m,n)*randn(n);
R = randn(m,n-1)*randn(n-1,n);

% Create some scalar
I = 24958734475;

% Print ranks of F, R, I*F, I*R
clc
disp( ['rank(F) = ' num2str(rank(F)) ])
disp( ['rank(R) = ' num2str(rank(R)) ])
disp(' ')
disp( ['rank(I*F) = ' num2str(rank(I*F)) ])
disp( ['rank(I*R) = ' num2str(rank(I*R)) ])

% Check whether rank(I*F) == I*rank(F) - true if rank is a linear operator
% False, rank is not a linear operator
disp(' ')
disp( ['rank(I*R) = ' num2str(rank(I*R)) ])
disp( ['I*rank(R) = ' num2str(I*rank(R)) ])
```

Results:

rank(F) = 4

rank(R) = 3

rank(I*F) = 4

rank(I*R) = 3

rank(I*R) = 3

I*rank(R) = 74876203425

71. RANK OF $A^T A$ AND $A A^T$

$$\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T \mathbf{A}) = \text{rank}(\mathbf{A} \mathbf{A}^T) = \text{rank}(\mathbf{A})$$

Why $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T \mathbf{A})$:

1. $\mathbf{A}^T \mathbf{A}$ spans the same space as \mathbf{A}
2. $\mathbf{A}^T \mathbf{A}$ has the same dimensionality as \mathbf{A}
3. $\mathbf{A}^T \mathbf{A}$ has the same singular values (squared) as \mathbf{A}

MATLAB

Code: linalg_matrixRank.m

Lines: 50 to 71

Keywords: rank, randn, round, fprintf, size

```
%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Matrix rank
% VIDEO: Rank of A^TA and AA^T
% Instructor: sincxpress.com
%
%%

% matrix sizes
m = 14;
n = 3;

% create matrices
A = round( 10*randn(m,n) );

AtA = A'*A;
AAAt = A*A';

fprintf('\n AtA: %gx%g, rank=%g',size(AtA,1),size(AtA,2),rank(AtA));
fprintf('\n AAAt: %gx%g, rank=%g\n',size(AAAt,1),size(AAAt,2),rank(AAAt));
```

Output:

AtA: 3x3, rank=3

AAAt: 14x14, rank=3

72. CODE CHALLENGE: RANK OF MULTIPLIED AND SUMMED MATRICES

MATLAB

Code: Lesson_70_Code_Challenge_Rank_Mult_Sum.m

Keywords: rand, num2str, disp, rank

```
%% 70. Code challenge: rank of multiplied and summed matrices
% Rules: rank of AtA*BtB<=min( rank(A),rank(BtB) )
% rank of AtA+BtB<=rank(AtA)+rank(BtB)
```

```

% Generate two rectangular random matrices (A and B), 2x5
m = 2;
n = 5;

A = rand(m,n);
B = rand(m,n);

% Find their ranks
disp(' ')
disp( ['rank(A) = ' num2str(rank(A))] )
disp( ['rank(B) = ' num2str(rank(B))] )

% Compute AtA and BtB
AtA = A'*A;
BtB = B'*B;

% find ranks of AtA and BtB
disp(' ')
disp( ['rank(AtA) = ' num2str(rank(AtA))] )
disp( ['rank(BtB) = ' num2str(rank(BtB))] )

disp(' ')
disp( ['rank(AtA * BtB) = ' num2str(rank(AtA * BtB))] )
disp( ['rank(AtA + BtB) = ' num2str(rank(AtA + BtB))] )

```

Output:

```

rank(A) = 2
rank(B) = 2

rank(AtA) = 2
rank(BtB) = 2

rank(AtA * BtB) = 2
rank(AtA + BtB) = 4

```

73. MAKING A MATRIX FULL-RANK BY “SHIFTING”

Full rank matrices are used in many analysis projects. Using shifting as a method to generate a full rank matrix from a reduced rank matrix.

In multivariate statistics and machine learning, shifting is also called: inflation, smoothing, regularization.

Reduced rank matrices may be caused by:

- Something went wrong with data collection
- System being measured is a low rank system
- Use a subspace system to reduce noise or separate multiple sources of signals

Definition of shifting a matrix:

$$\tilde{\mathbf{A}} = \mathbf{A} + \lambda \mathbf{I}$$

Setting the value of λ is specific to each situation. The goal is to make the matrix rank full while changing the values within the matrix as little as possible.

“Shifting” a matrix: normal example

$$\begin{pmatrix} 1 & 3 & -19 \\ 5 & -7 & 59 \\ -5 & 2 & -24 \end{pmatrix}_{r=2} + .01 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}_{r=3} = \begin{pmatrix} 1.01 & 3 & -19 \\ 5 & -6.99 & 59 \\ -5 & 2 & -23.99 \end{pmatrix}_{r=3}$$

BiophysicsLab.com

Figure 66 - "Shifting" a matrix: normal example

MATLAB

Code: linalg_matrixRank.m

Lines: 72 to 99

Keywords: imagesc, rank, round, eye

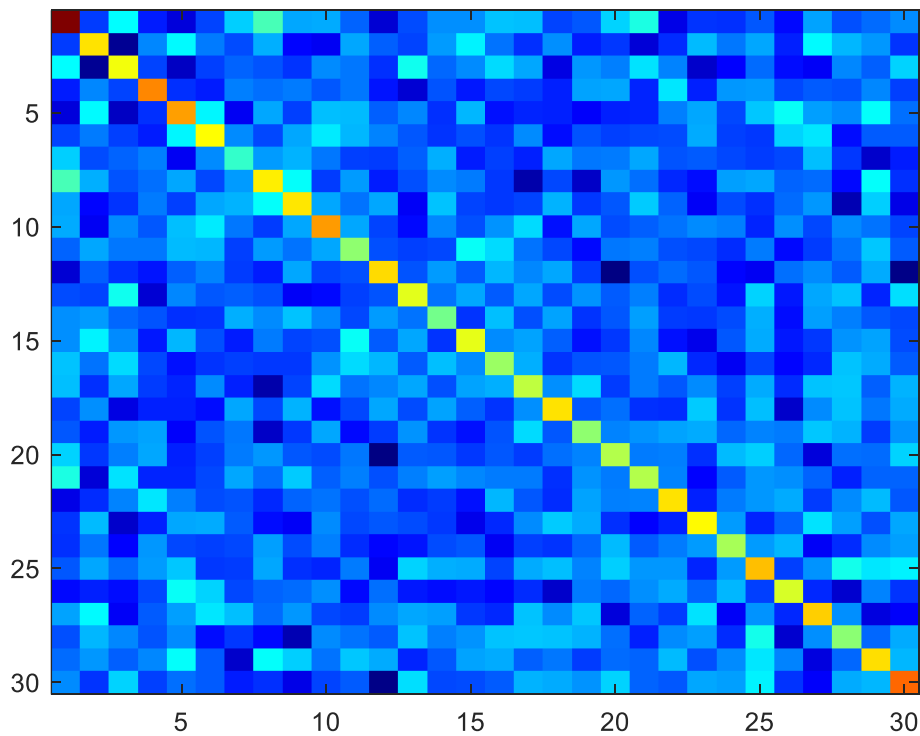


Figure 67 - View a symmetric matrix with imagesc

Code output:

rank(w/o shift) = 29

rank(with shift) = 30

74. CODE CHALLENGE: IS THIS VECTOR IN THE SPAN OF THE SET

MATLAB

Code: Lesson_72_Code_Challenge_Vector_in_Set.m

Keyword: rank

```
%% Lesson 72. Code challenge: is this vector in the span of the set?

% Determine whether this column vector
v = [1 2 3 4]';

% Is in the span of these sets
% - algebraic definition: Create a linearly dependent set of vectors
% - geometric definition: Vector v does not expand the dimensionality
S = [ 4 3 6 2]' [0 4 0 1]';
T = [ 1 2 2 2]' [0 0 1 2]';

% Create an augmented matrix - the rank will not change if v is in the set
```

rank([S v]) % v is not in the span of S because rank increased by 1
 rank([T v]) % v is in the span of T because the rank has not changed

SECTION 7: MATRIX SPACES

See included exercises with code download:

- linalg_matrixSpaces.ipynb
- linalg_matrixSpaces.m
- linalg_matrixSpaces.pdf

77. COLUMN SPACE OF A MATRIX

Important prerequisites:

- [Vector spaces](#)
- [Span](#)
- [Linear independence](#)
- [Basis](#)

Notation: $C(\mathbf{A})$

Definition: the vector subspace spanned by the columns of \mathbf{A}

$$C(\mathbf{A}) = \{\beta_1 \mathbf{a}_1 + \dots + \beta_n \mathbf{a}_n\}; \quad \beta \in \mathbb{R}$$

$$= \text{span}(\{\mathbf{a}_1, \dots, \mathbf{a}_n\})$$

Important question: $\mathbf{v} \in C(\mathbf{A})$?

$$\begin{pmatrix} -3 \\ 1 \\ 5 \end{pmatrix} \in C\left(\begin{pmatrix} 3 & 0 \\ 5 & 2 \\ 1 & 2 \end{pmatrix}\right) \quad \left(-1 \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix} + 3 \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}\right) = \begin{pmatrix} -3 \\ 1 \\ 5 \end{pmatrix}$$

BiophysicsLab.com

Figure 68 – Verify vector is in column space of matrix

If vector is in the column space of a matrix:

- Yes: What are the coefficients
- No: How close is it to $C(\mathbf{A})$

Important question: $\mathbf{v} \in C(\mathbf{A})$?

$$\begin{pmatrix} 1 \\ 7 \\ 3 \end{pmatrix} \notin C \left(\begin{pmatrix} 0 & 0 \\ 5 & 2 \\ 1 & 2 \end{pmatrix} \right) \Rightarrow \left\| \begin{pmatrix} 0 & 0 \\ 5 & 2 \\ 1 & 2 \end{pmatrix} \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} - \begin{pmatrix} 1 \\ 7 \\ 3 \end{pmatrix} \right\| = \left\| \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} \right\|$$

BiophysicsLab.com

Figure 69 - Determine how close vector is to column space of matrix

The magnitudes of the vectors in figure 67 will always be greater than zero. Minimize the equation finding the right W values to minimize the values of V. This problem is solved using Least Squares. The most important algorithm used in statistics and machine learning.

78. COLUMN SPACE, VISUALIZED IN CODE

MATLAB

Code: linalg_matrixSpaces.m

Lines: 1 to 32

Keywords: figure, clf, plot3, ezmesh, set, xlabel, ylabel, zlabel, title, axis grid, rotate3d

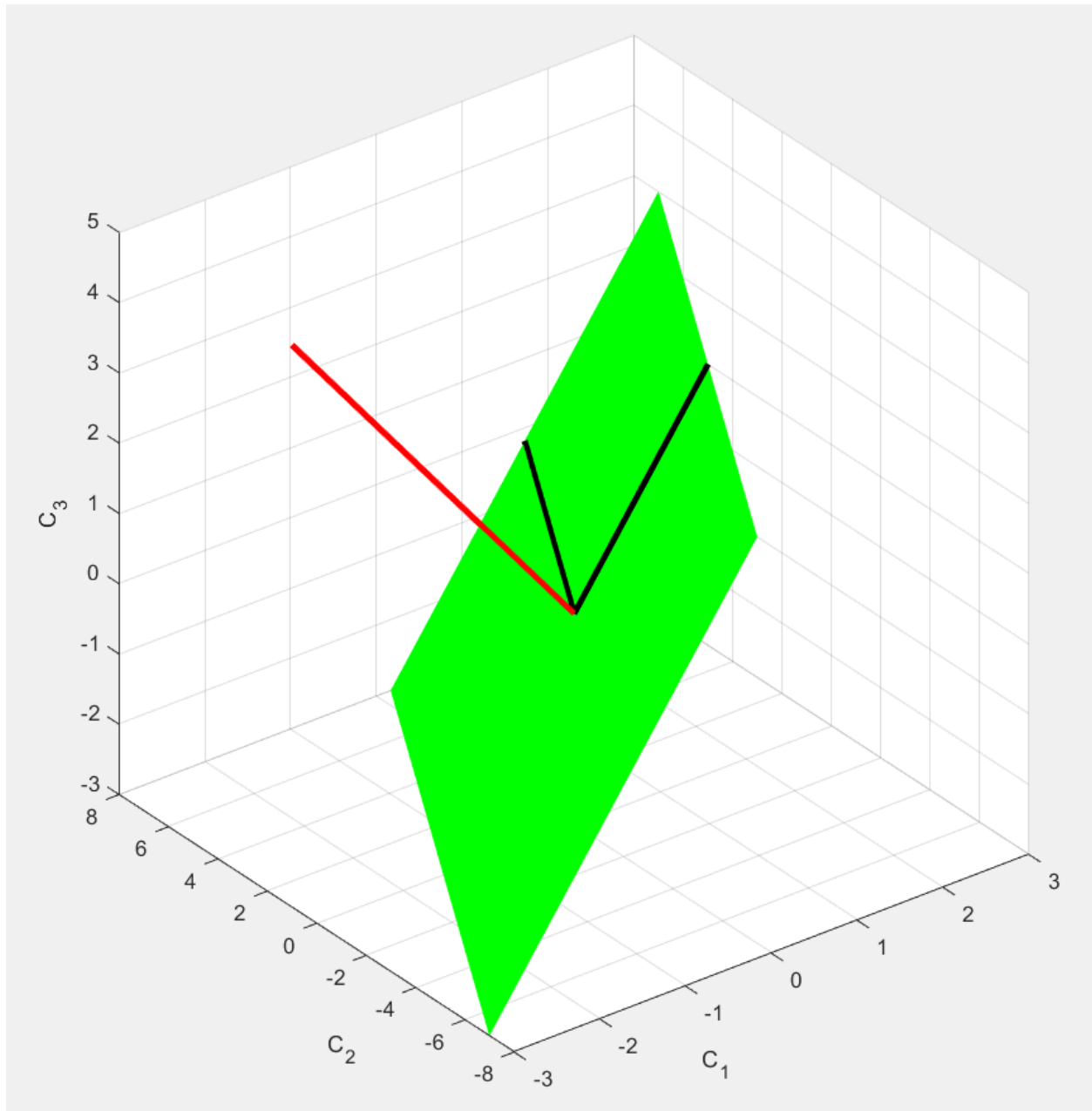


Figure 70 – MATLAB demonstration of a vector v (red) contained in S (green plane)

79. ROW SPACE OF A MATRIX

Notation: $R(A)$

$C(A^T)$

Rows vs. columns: example application

Assume sensors are layout out horizontally and time points increase vertically, then

- $R(A)$: combinations of time points (e.g., temporal filter)

- $C(\mathbf{A})$: combinations of sensors (e.g., spatial filter)

Column space

$$\begin{pmatrix} 3 & 0 \\ 5 & 2 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ 3 \end{pmatrix} = \begin{pmatrix} -3 \\ 1 \\ 5 \end{pmatrix}$$

Row space

$$\begin{pmatrix} -1 & 0 & 3 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 5 & 2 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 6 \end{pmatrix}$$

BiophysicsLab.com

Figure 71 - Column space vs. Row space

80. NULL SPACE AND LEFT NULL SPACE OF A MATRIX

For some matrices it is possible to find a nontrivial vector that can be multiplied by a matrix to create a zero vector. For other matrices where it is not possible, the null space is empty.

Definition of a **null space** for a matrix:

$N(\mathbf{A})$: The set of all vectors $\{\mathbf{v}\}$ such that $\mathbf{A}\mathbf{v} = \mathbf{0}$ and $\mathbf{v} \neq \mathbf{0}$

A matrix with no null space (or empty null space) has linearly independent columns.

If a matrix has a linearly independent set of columns then the matrix has the empty set for the null space.

The null space is used in Eigen decomposition.

Null space of a matrix: relation to independence

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}$$

$$r(\mathbf{A}) = 2$$

$$N(\mathbf{A}) = \{ \}$$

$$\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$$

$$r(\mathbf{A}) = 1$$

$$N(\mathbf{A}) = \left\{ \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\}$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$r(\mathbf{A}) = 0$$

$$N(\mathbf{A}) = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

BiophysicsLab.com

Figure 72 - Null space of a matrix: relation to independence

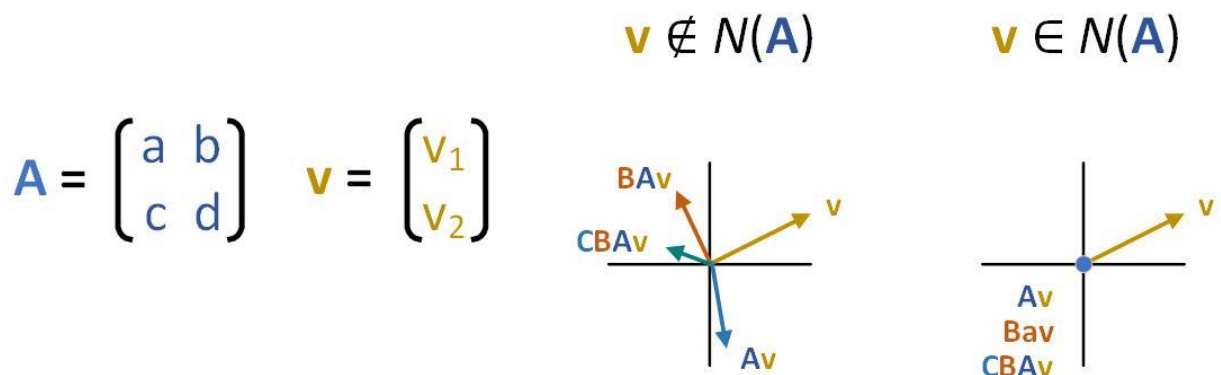
Left null space (involving row vectors and matrix rows)

Definition:

$N(\mathbf{A}^T)$: The set of all vectors $\{\mathbf{v}\}$ such that $\mathbf{v}^T \mathbf{A} = \mathbf{0}^T$ and $\mathbf{v}^T \neq \mathbf{0}^T$

Also written: $\mathbf{A}^T \mathbf{v} = \mathbf{0}$

Null space of a matrix: interpretation



BiophysicsLab.com

Figure 73 - Null space of a matrix: Graphic interpretation

The figure above shows the relationship between null space and linear independence of the columns of a matrix.

81. COLUMN/LEFT-NULL AND ROW/NULL SPACES ARE ORTHOGONAL

The column space is orthogonal to the left null space.

The row space is orthogonal to the null space.

82. DIMENSIONS OF COLUMN/ROW/NULL SPACES

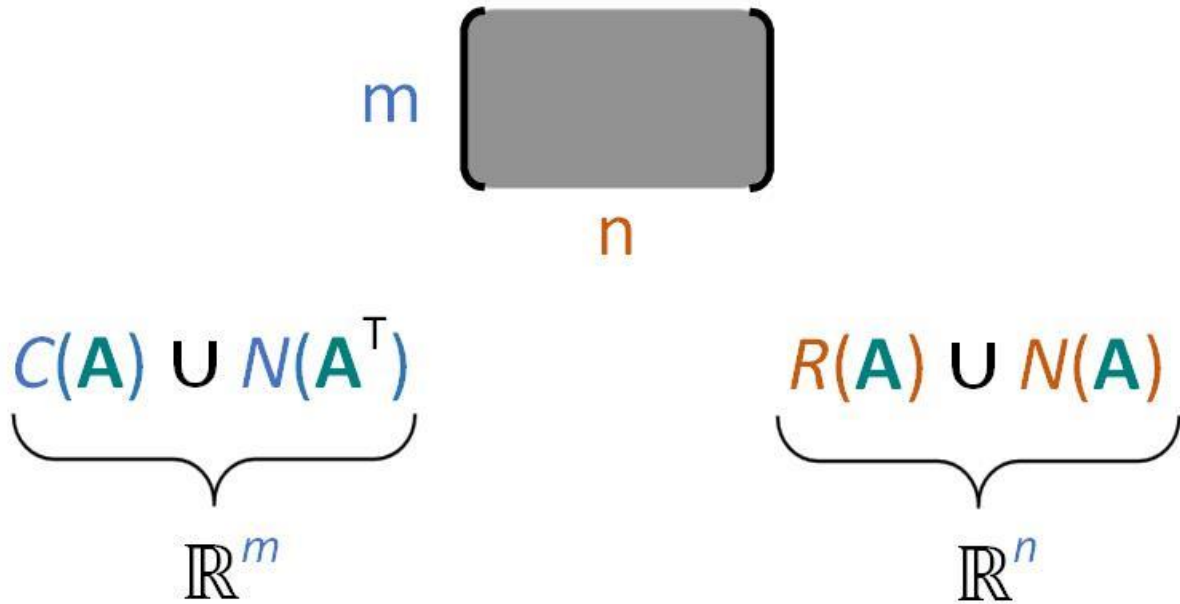
Orthogonal subspaces have complementary dimensionalities that sum to M (column + left-null) or N (row + null).

83. EXAMPLE OF THE FOUR SUBSPACES

Referring to the figure below “Picture of the four subspaces” for an $m \times n$ matrix:

- The union of the column space of the matrix and the left null space form orthogonal complements. Any column vector that is in the column space of A is orthogonal to any vector that is in the left null space of A . These two subspaces taken together span all of \mathbb{R}^m . So, m in \mathbb{R}^m is the sum of the dimensionality of column space plus the dimensionality of the left null space.
- The union of the row subspace of A with the null subspace of A together produces all of \mathbb{R}^n . This union also forms orthogonal complements. So, any vector in the row space will be orthogonal to any vector in the null space. The dimensionality of the row space plus the dimensionality of the null space adds up to n .

Picture of the four subspaces



BiophysicsLab.com

Figure 74 - Picture of the four subspaces

Example of span as one of the four subspaces

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 3 & 3 \end{pmatrix}$$

$$C(\mathbf{A}) = \left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\} \in \mathbb{R}^2 \quad \dim(C(\mathbf{A}))=2$$

$$C(\mathbf{A}) = \text{span} \left[\left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\} \right]$$

$$C(\mathbf{A}) = \lambda \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \beta \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad \lambda, \beta \in \mathbb{R}$$

BiophysicsLab.com

Figure 75 - Example of span as one of the four subspaces

Example of the four subspaces

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 3 & 3 \end{pmatrix}$$

$$C(\mathbf{A}) = \left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\} \in \mathbb{R}^2 \quad \dim(C(\mathbf{A}))=2$$

$$R(\mathbf{A}) = \left\{ \begin{bmatrix} 1 \\ 3 \end{bmatrix}^T, \begin{bmatrix} 2 \\ 3 \end{bmatrix}^T \right\} \in \mathbb{R}^3 \quad \dim(R(\mathbf{A}))=2$$

$$N(\mathbf{A}^T) = \left\{ \right\} \in \mathbb{R}^2 \quad \dim(N(\mathbf{A}^T))=0$$

$$N(\mathbf{A}) = \left\{ [2 \ -1 \ 1]^T \right\} \in \mathbb{R}^3 \quad \dim(N(\mathbf{A}))=1$$

BiophysicsLab.com

Figure 76 - Example of the four subspaces

$Ax = b$: Is there an exact solution?

$$Ax = b$$

Solution when $b \in \mathcal{C}(A)$

$$x_1 \begin{bmatrix} a_{1,1} \\ a_{2,1} \end{bmatrix} + x_2 \begin{bmatrix} a_{1,2} \\ a_{2,2} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$Ax = \hat{b}$$

Solution when $\hat{b} \in \mathcal{C}(A)$

$$\|\hat{b} - b\|$$

BiophysicsLab.com

Figure 77 - $Ax=b$: Is there an exact solution?

$$Ax = 0$$

$$(A - \lambda I)x = 0$$

Where:

λ is an Eigenvalue

x is an Eigenvector

SECTION: 8: SOLVING SYSTEMS OF EQUATIONS

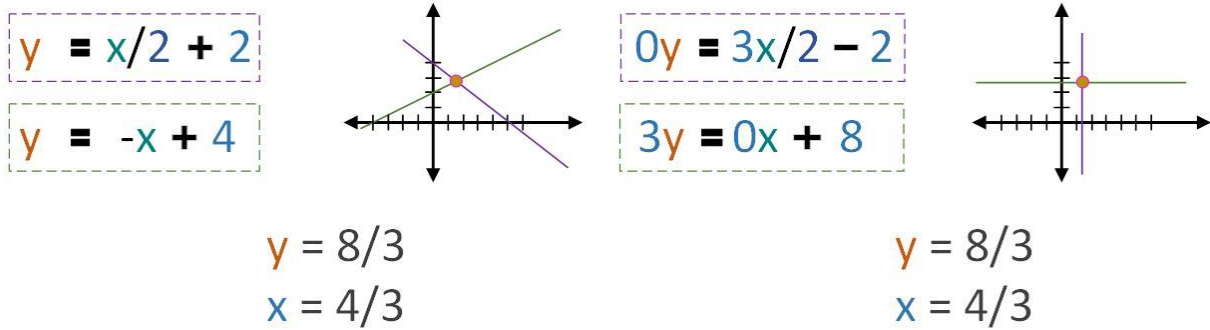
See included exercises with code download:

- linalg_matrixSystems.pdf
- linalg_systems.ipynb
- linalg_systems.m

86. SYSTEMS OF EQUATIONS: ALGEBRA AND GEOMETRY

There is a relationship between systems of equations and their geometric equivalence.

Systems of equations: algebra and geometry



BiophysicsLab.com

Figure 78 - Systems of equations: algebra and geometry

Multiply top left equation by 2, add that result to the left bottom equation, to generate equation at bottom right.

Subtract bottom left equation from top left equation, to generate equation at top right.

The right plot is an **orthogonalization** of the left system of equations. The plot lines move around, but the point of intersection stays the same. This same process is used in **Eigendecomposition**.

MATLAB

Code: linalg_systems.m

Lines: 1 to 87

Keywords: randn, fplot, plot, pause, figure, get, gca, cla, clf, fplot3, repmat, set, axis, rotate3d

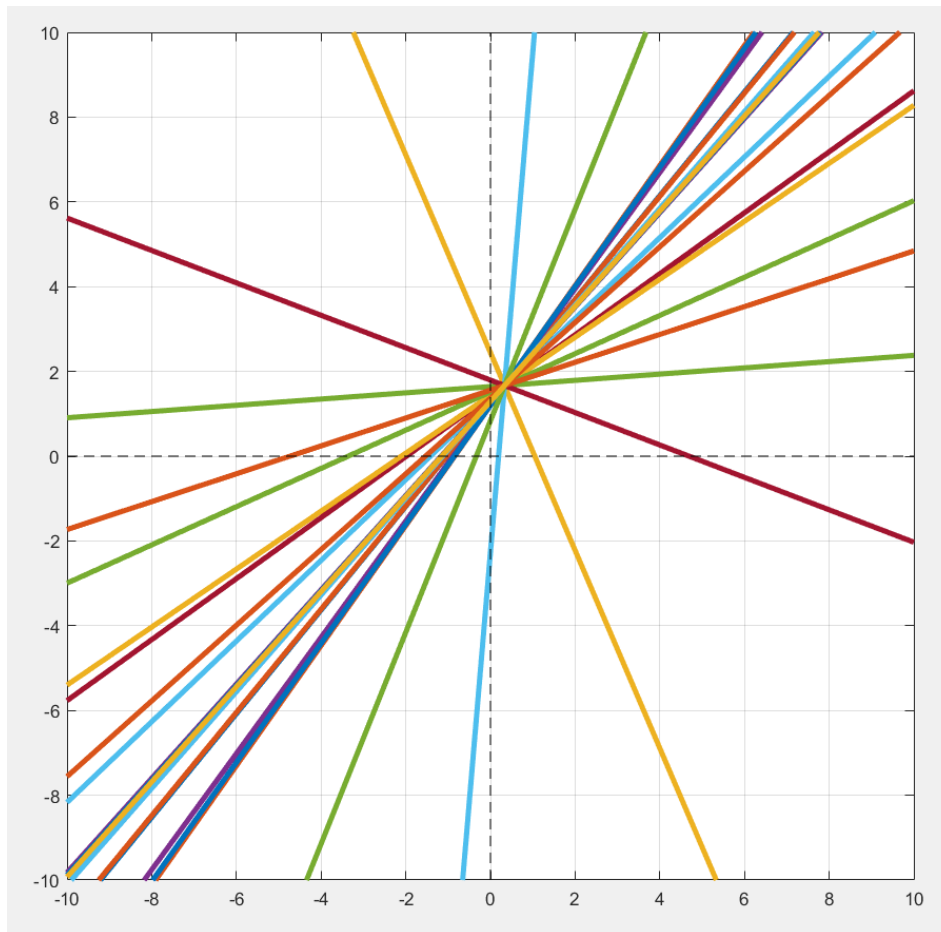


Figure 79 - MATLAB 2d linear system of equations demonstration

Figure above shows that all variants of two basis vectors converge at one point.

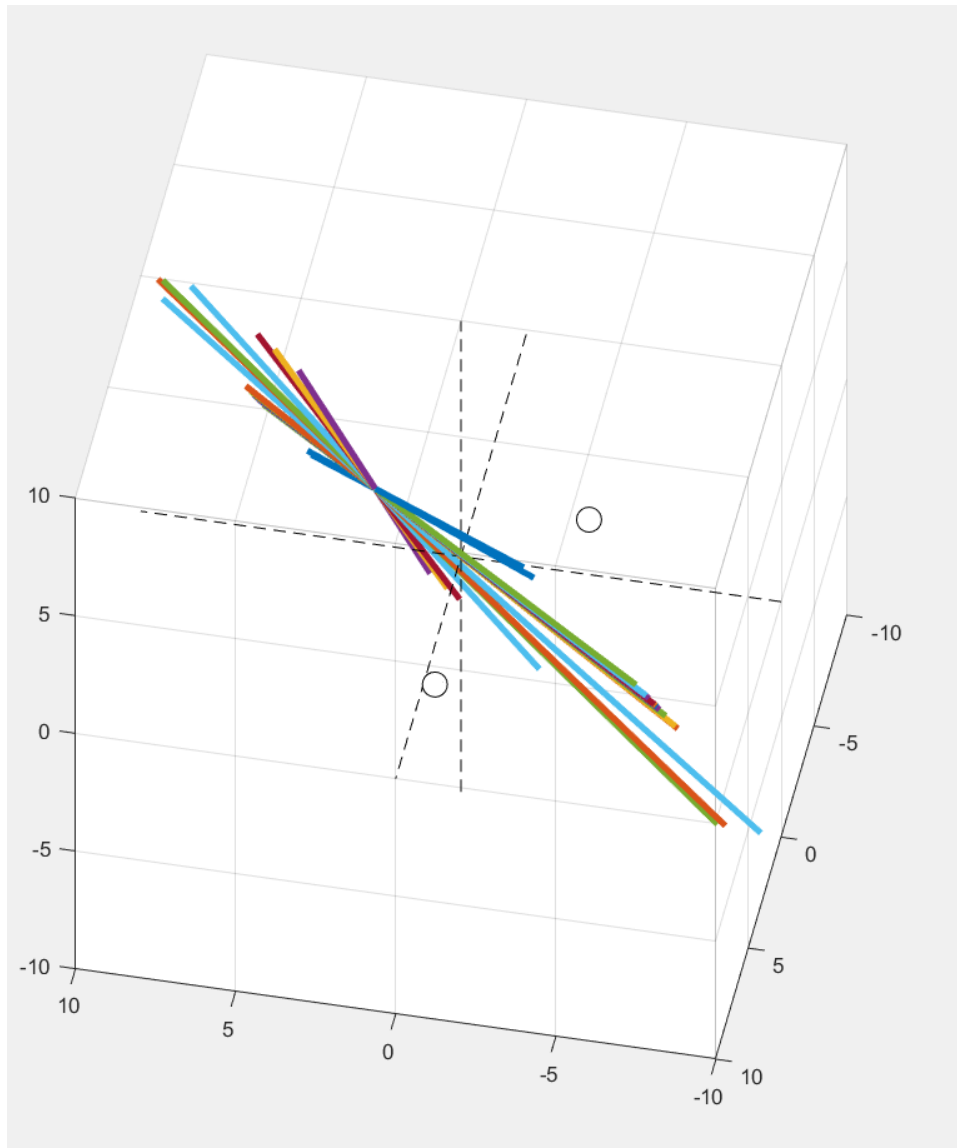


Figure 80 - MATLAB 3d linear system of equations demonstration

Figure above shows that all variants of two basis vectors converge at one point in 3D. In addition, all lines are in one plane.

87. CONVERTING SYSTEMS OF EQUATIONS TO MATRIX EQUATIONS

Converting systems of equations to matrix equations

$$\begin{aligned}2x + 3y - 4z &= 5 \\ -x + 9z &= 7\end{aligned}$$

$$\begin{pmatrix} 2 & 3 & -4 \\ -1 & 0 & 9 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

$2 \times 3 \qquad 3 \times 1 \qquad 2 \times 1$

BiophysicsLab.com

Figure 81 - Converting systems of equations to matrix equations

88. GAUSSIAN ELIMINATION

6-Step plan for solving a system of equations via matrix methods:

1. Convert system to matrix-vector equation (see lesson 85)

Step 1: Convert system of equations to matrices

$$\begin{aligned}2x + 3y &= 5 \\ -x + 7y &= 7\end{aligned}$$
$$\begin{pmatrix} 2 & 3 \\ -1 & 7 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

BiophysicsLab.com

Figure 82 - Convert system of equations to matrices

2. Augment the coefficients matrix with the vector of constants (see lesson 33)

Step 2: Augment coefficients matrix with constants

$$\left[\begin{array}{cc|c} 2 & 3 & 5 \\ -1 & 7 & 7 \end{array} \right]$$

BiophysicsLab.com

Figure 83 - Augment coefficients matrix with constants

- Use Gaussian elimination to create a matrix with 1's on diagonals (see lesson 65 Echelon form)

Step 3: Convert matrix into an upper-triangular matrix

$$\begin{bmatrix} 2 & 3 & | & 5 \\ -1 & 7 & | & 7 \end{bmatrix} \xrightarrow{\frac{1}{2} \times \text{Row}_1} \begin{bmatrix} 1 & 3/2 & | & 5/2 \\ -1 & 7 & | & 7 \end{bmatrix} \xrightarrow{\text{Row}_1 + \text{Row}_2} \begin{bmatrix} 1 & 3/2 & | & 5/2 \\ 0 & 17/2 & | & 19/2 \end{bmatrix} \xrightarrow{2/17 \times \text{Row}_2} \begin{bmatrix} 1 & 3/2 & | & 5/2 \\ 0 & 1 & | & 19/17 \end{bmatrix}$$

BiophysicsLab.com

Figure 84 - Convert matrix into an upper-triangular matrix

- Map the matrix back to the equations

Step 4: Map matrix back onto equations

$$\begin{aligned} 2x + 3y &= 5 \\ -x + 7y &= 7 \end{aligned} \xrightarrow{\quad} \begin{bmatrix} 1 & 3/2 & | & 5/2 \\ 0 & 1 & | & 19/17 \end{bmatrix}$$

$$\begin{aligned} 1x + 3/2y &= 5/2 \\ 0x + 1y &= 19/17 \end{aligned}$$

BiophysicsLab.com

Figure 85 - Map matrix back onto equations

- Back-substitution to find solutions

Step 5: Map matrix back onto equations

$$\begin{aligned} x + 3/2y &= 5/2 \\ y &= 19/17 \end{aligned}$$

$$\begin{aligned} x + 57/34 &= 5/2 \\ x &= 28/34 = 14/17 \end{aligned}$$

BiophysicsLab.com

Figure 86 - Back-substitution to solve for variables

- Check your work using the original system

Step 6: Check with the original system

$$\begin{aligned} y &= 19/17 \\ x &= 14/17 \end{aligned}$$

$$\begin{aligned} 2x + 3y &= 5 \\ -x + 7y &= 7 \end{aligned} \xrightarrow{\quad} \begin{aligned} 28/17 + 57/17 &= 5 \\ -14/17 + 133/17 &= 7 \end{aligned}$$

BiophysicsLab.com

Other possibilities include a reduced rank, or rank deficient, matrix where one or more equations is a multiple of another equation in the matrix. This possibility makes the solution set smaller. See figure below.

This method can be used to determine the rank of a matrix.

Other possibilities: reduced rank

$$\left(\begin{array}{cccc|c} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \end{array} \right)$$

BiophysicsLab.com

An impossible situation, or inconsistent result, occurs when a system of equations in echelon form has a row of zeros equaling some number. The system does not have a single solution. Geometrically the lines described by these equations never meet at a single point. See figure below.

Other possibilities: inconsistent

$$\left(\begin{array}{cccc|c} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & * \end{array} \right)$$

BiophysicsLab.com

Echelon form and pivots

$$\begin{pmatrix} \boxed{a} & b & c & d & e \\ 0 & \boxed{f} & g & h & i \\ 0 & 0 & 0 & \boxed{j} & k \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

BiophysicsLab.com

Figure 87 - Echelon form and pivots

Echelon form: zeros are below and to the left of the pivots in the matrix. The leading non-zero terms are called pivots. In figure above the pivots have boxes around them. Pivots do not need to be in the column to the right and below the previous pivot.

The rank of a matrix is also the number of pivots of a matrix in its echelon form.

Pivot values do not need to be “1” as in the Gaussian elimination technique described in the previous lesson. Also there is no extra “solution” vector augmenting the matrix as in the Gaussian elimination technique.

Tips for putting a matrix into echelon form:

- Use the first row to knock out subsequent rows because the first row should have its pivot value in the first column.
- Useful to try and eliminate the bottom row first. Then work your way up the rows.

Echelon form conversion: example

$$\begin{pmatrix} 1 & 2 & 2 \\ -1 & 3 & 0 \\ 2 & 4 & 4 \end{pmatrix} \xrightarrow{-2R_1+R_3} \begin{pmatrix} 1 & 2 & 2 \\ -1 & 3 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{R_1+R_2} \begin{pmatrix} 1 & 2 & 2 \\ 0 & 5 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

BiophysicsLab.com

Figure 88 - Echelon form conversion: example

Echelon form has several primary uses:

- Computing rank of a matrix.
- Working towards Gaussian elimination
- Stepping stone to the reduced row echelon form

90. REDUCED ROW ECHELON FORM

RREF or $\text{rref}(\mathbf{A})$ abbreviation for Row Reduced Echelon Form

Information is lost when matrix is in RREF. Keep a record of transformations made so an inverse matrix can be generated.

Any square matrix has the identity matrix as its RREF.

Reduced row echelon form: examples

$$\begin{pmatrix} 1 & 0 & * & 0 & 0 \\ 0 & 1 & * & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & * & * & 0 & 0 \\ 0 & 1 & * & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & * & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

BiophysicsLab.com

Figure 89 - Reduced row echelon form: examples

In the figure above, only the matrix to the far left is in reduced row echelon form.

Reduced row echelon form: worked problem

$$\begin{array}{c}
 \begin{pmatrix} 1 & 2 & 4 & 5 \\ 2 & 4 & 5 & 4 \\ 4 & 5 & 4 & 2 \end{pmatrix} \xrightarrow{-2R_1+R_2} \begin{pmatrix} 1 & 2 & 4 & 5 \\ 0 & 0 & -3 & -6 \\ 4 & 5 & 4 & 2 \end{pmatrix} \xrightarrow{-4R_1+R_3} \begin{pmatrix} 1 & 2 & 4 & 5 \\ 0 & 0 & -3 & -6 \\ 0 & -3 & -12 & -18 \end{pmatrix} \\
 \begin{matrix} -1/3R_2 \rightarrow \\ -1/3R_3 \rightarrow \end{matrix} \begin{pmatrix} 1 & 2 & 4 & 5 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 4 & 6 \end{pmatrix} \xrightarrow{\text{Swap } R_2 \text{ \& } R_3} \begin{pmatrix} 1 & 2 & 4 & 5 \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 1 & 2 \end{pmatrix} \xrightarrow{-2R_2+R_1} \begin{pmatrix} 1 & 0 & -4 & -7 \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 1 & 2 \end{pmatrix} \\
 \xrightarrow{4R_3+R_1} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 4 & 6 \\ 0 & 0 & 1 & 2 \end{pmatrix} \xrightarrow{-4R_3+R_2} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 2 \end{pmatrix}
 \end{array}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}$$

BiophysicsLab.com

Figure 90 - Reduced row echelon form: worked problem

MATLAB

Code: linalg_matrixSpaces.m

Lines: 88 to 111

Keywords: randn, rref

```
ans =
     1     2     4     5     1     0     0     1
     2     4     5     4     0     1     0    -2
     4     5     4     2     0     0     1     2
```

Rref() function produces the same result as the figure above, Reduced row echelon form: worked problem.

91. CODE CHALLENGE: RREF OF MATRICES WITH DIFFERENT SIZES AND RANKS

MATLAB

Code: Lesson_89_Code_Challenge_RREF_Different_Sizes_Ranks.m

Keywords: randn, rref

```

%% Lesson 89. Code challenge: RREF of matrices with different sizes and ranks
% - square
% - rectangular (tall, wide)
% - linear dependencies (columns, rows)

% square: produces identity matrix
m = 5;
n=5;
A = randn(m,n);
rref(A)

% rectangular
% tall: produces reduced rank matrix
% wide: not sure what is produced?
m = 8;
n = 3;
A = randn(m,n);
rref(A)
rref(A')

% Linear dependencies
% Redundant column reduces rank
m = 5;
n=5;
A = randn(m,n);
A(:,1) = A(:,2);
rref(A)
% Redundant column
m = 5;
n=5;
A = randn(m,n);
A(1,:) = A(2,:);
rref(A)

```

92. MATRIX SPACES AFTER ROW DEDUCTION

This section discusses rank, row spaces and column space of a matrix.

Rank is a property of the entire Matrix. Rank does not change after row reduction.

Row space does not change RREF.

Column space does change during row reduction. Each change to an entire row during RREF changes one or more numbers in a column, but not an entire column.

MATLAB

Code: linalg_matrixSpaces.m

Lines: 112 to 153

Keywords: rref, figure, clf, hold, ezmesh, set, norm, plot3, xlabel, ylabel, zlabel, asix, grid, title, legend

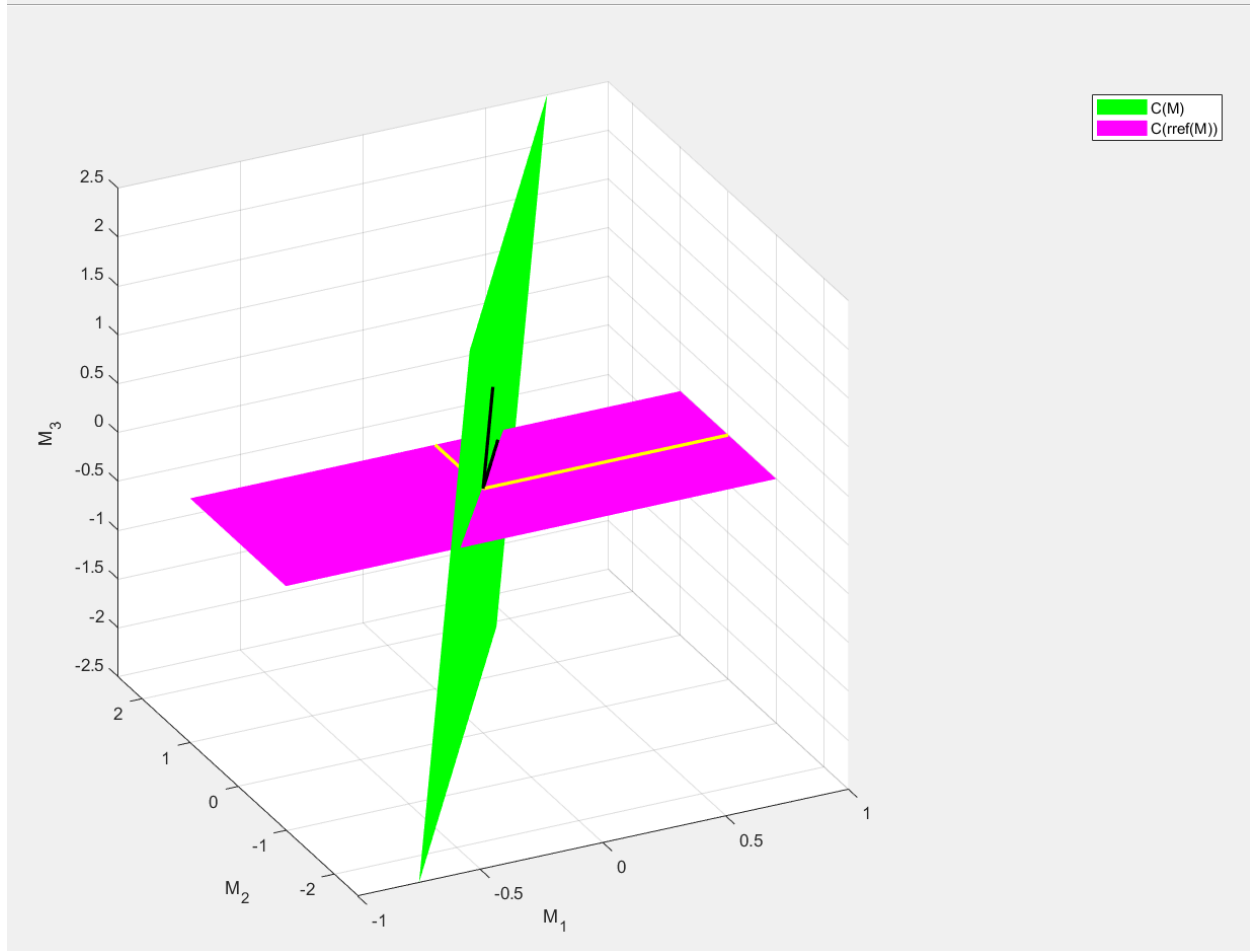


Figure 91 - MATLAB Matrix spaces after row reduction

SECTION 9: MATRIX DETERMINANT

See included exercises with code download:

- linalg_matrixDet.pdf
- no MATLAB
- no python code

94. DETERMINANT: CONCEPT AND APPLICATIONS.

Concept:

1. Notation: $\det(A)$ or $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$
2. Only square matrices have a determinant.
3. The determinant is a single number that reflects the entire matrix.

4. $\det(A)=0$ (a.k.a singular or non-invertible) if linearly dependent columns

Applications:

1. Geometry: Area/volume of shape specified by coordinates in the matrix.
2. Matrix inverse: divide by determinant (no inverse if determinant is zero)

95. DETERMINANT OF A 2X2 MATRIX

The determinant of the identity matrix is always 1

$$\det(\mathbf{I}) = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1-0$$

$$\det(\mathbf{A}) = \begin{vmatrix} c & d \\ a & b \end{vmatrix} = bc-ab = -(ad-bc), \text{ where swapping rows only changes the sign.}$$

96. CODE CHALLENGE: DETERMINANT OF SMALL AND LARGE SINGULAR MATRICES

MATLAB

Code: Lesson_95_Code_Challenge_RREF_Different_Sizes_Ranks.m

Keywords: det, randn, help

```
%% Lesson 94. Code challenge: determinant of small and large singular matrices

% singular matrix (reduced-rank matrix) has a determinant of 0

% generate a 2x2 matrix of integers, with linear dependencies
% compute the rank
A = [1 3; 1 3];
det(A)

% generate mxm matrices, impose linear dependencies
% compute the rank
% small m and for large m
m = 2;
A = randn(m);
det(A) % result will be non zero

A(:,1) = A(:,2);
det(A) % result will be zero

m = 15;
A = randn(m);
A(:,1) = A(:,2);
det(A) % result is small but non zero
% "help det" states that "cond" should be used instead of "det" for matrix
% singularity.
rank(A) % rank will be 14 since one row is duplicated
```

Determinant of a 3x3 matrix: visual trick #1

3x3 determinant to solve

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$$

Two copies of augmented determinant with diagonals

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \quad \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$$

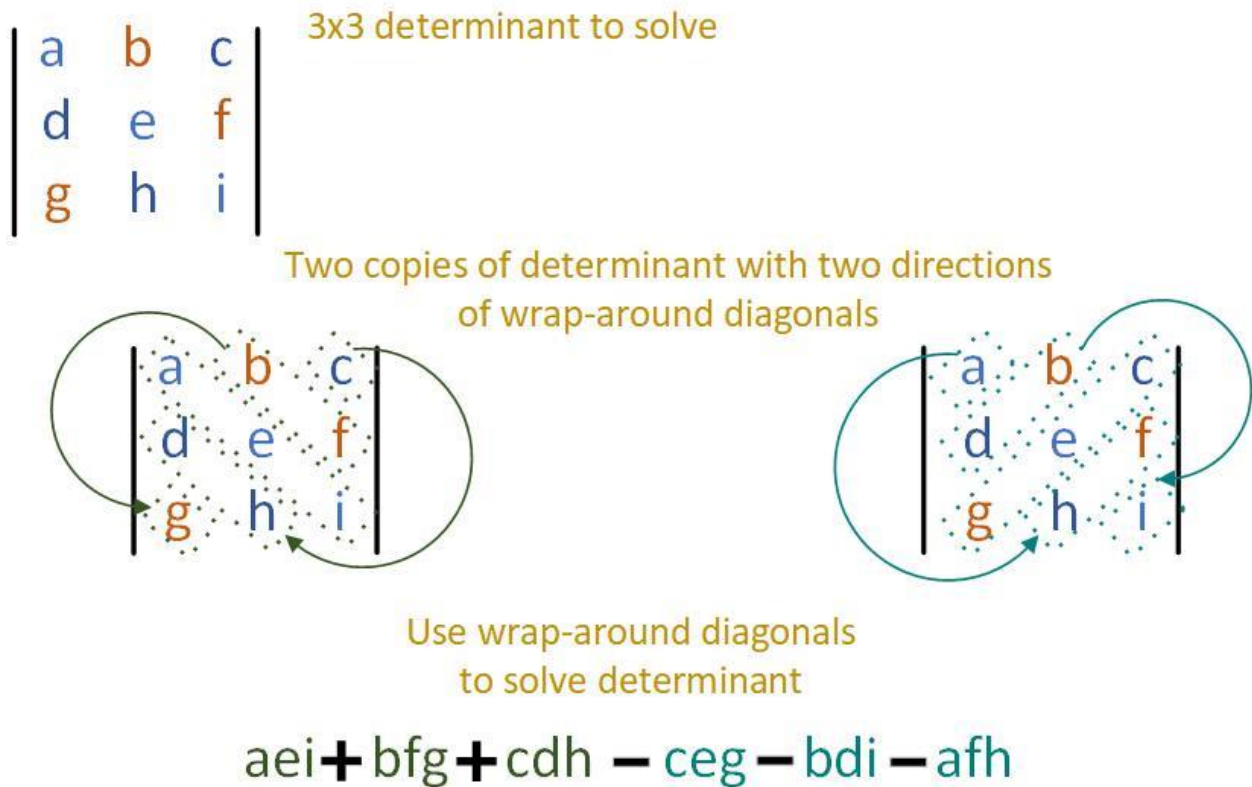
Use diagonals to solve determinant

$$aei + bfg + cdh - ceg - bdi - afh$$

BiophysicsLab.com

Figure 92 - Determinant of a 3x3 matrix: visual trick #1

Determinant of a 3x3 matrix: visual trick #2



BiophysicsLab.com

Figure 93 - Determinant of a 3x3 matrix: visual trick #2

If one column contains a scalar multiple of another column in a 3x3 matrix, then the determinant will be zero.

The identity matrix will always have a determinant value of one.

If one row in a 3x3 matrix is exchanged with another row, the determinant's magnitude is preserved but the sign is flipped. When two rows in a matrix are exchanged, both the determinant's magnitude and sign are preserved.

98. CODE CHALLENGE: LARGE MATRICES WITH ROW EXCHANGES

MATLAB

Code: Lesson_96_Code_Challenge_Large_Matrices_Row_Exchanges.m

Keywords: clc, randn, disp, det

```
%% Lesson 96 Code challenge: large matrices with row exchanges
```

```
% generate a 6x6 matrix:  
% - compute the determinant  
% - swap one row, det again  
% - swap two rows, det again
```

```
clc
```

```
A = randn(6);  
disp(['Before row swap: ' num2str(det(A)) ])
```

```
As = A([2 1 3 4 5 6],:); % swap rows 1 and 2  
disp(['After one row swap: ' num2str(det(As)) ])
```

```
As2 = A([2 1 3 5 4 6],:); % swap rows 1 and 2  
disp(['After two row swaps: ' num2str(det(As2)) ])
```

Command Window

```
Before row swap: -4.9517  
>> As = A([2 1 3 4 5 6],:); % swap rows 1 and 2  
disp(['After one row swap: ' num2str(det(As)) ])  
After one row swap: 4.9517  
>> As2 = A([2 1 3 5 4 6],:); % swap rows 1 and 2  
disp(['After two row swaps: ' num2str(det(As2)) ])  
After two row swaps: -4.9517
```

99. FIND MATRIX VALUES FOR A GIVEN DETERMINANT

Example:

Given

$$\begin{vmatrix} 5 - \lambda & -\frac{1}{3} \\ -3 & 5 - \lambda \end{vmatrix} = 0$$

Solve for λ

$$(5 - \lambda)(5 - \lambda) - 1 = 0$$

$$\lambda^2 + 25 - 10\lambda - 1 = 0$$

$$\lambda^2 - 10\lambda + 24 = 0$$

$$(\lambda - 6)(\lambda - 4) = 0$$

$$\lambda = 6, \lambda = 4$$

100. CODE CHALLENGE: DETERMINANT OF SHIFTED MATRICES

Code: Lesson_98_Code_Challenge_Determinant_Shifted_Matrices.m

Keywords: linspace, zeros, length, abs, mean, det, plot, xlabel, ylabel

```
%% Lesson_98_Code_Challenge_Determinant_Shifted_Matrices.m

% generate a square random matrix (20x20)
% impose a linear dependence
% "shift" the matrix (0->.1 times the identity matrix) (lambda)
% compute the size of the determinant of the shifted matrix: abs(determinant)
% repeat 1000 times, take the average abs(det)
% plot of det as a function of lambda
%

lambdas = linspace(0,.1,30);

% initialize the determinant vector
dets = zeros(length(lambdas),1);

for deti=1:length(lambdas)
    % run 1000 iterations
    for i=1:1000
        % generate a matrix
        M = randn(20);
        % reduce the rank (first column will be same as second column)
        M(:,1) = M(:,2);
        % compute the determinant
        tmp(i) = abs( det( M + lambdas(deti)*eye(20) ) );
    end

    dets(deti) = mean(tmp);
end

plot(lambdas, dets, 's-')
xlabel('Fraction of identity matrix for shifting')
ylabel('determinant')
```

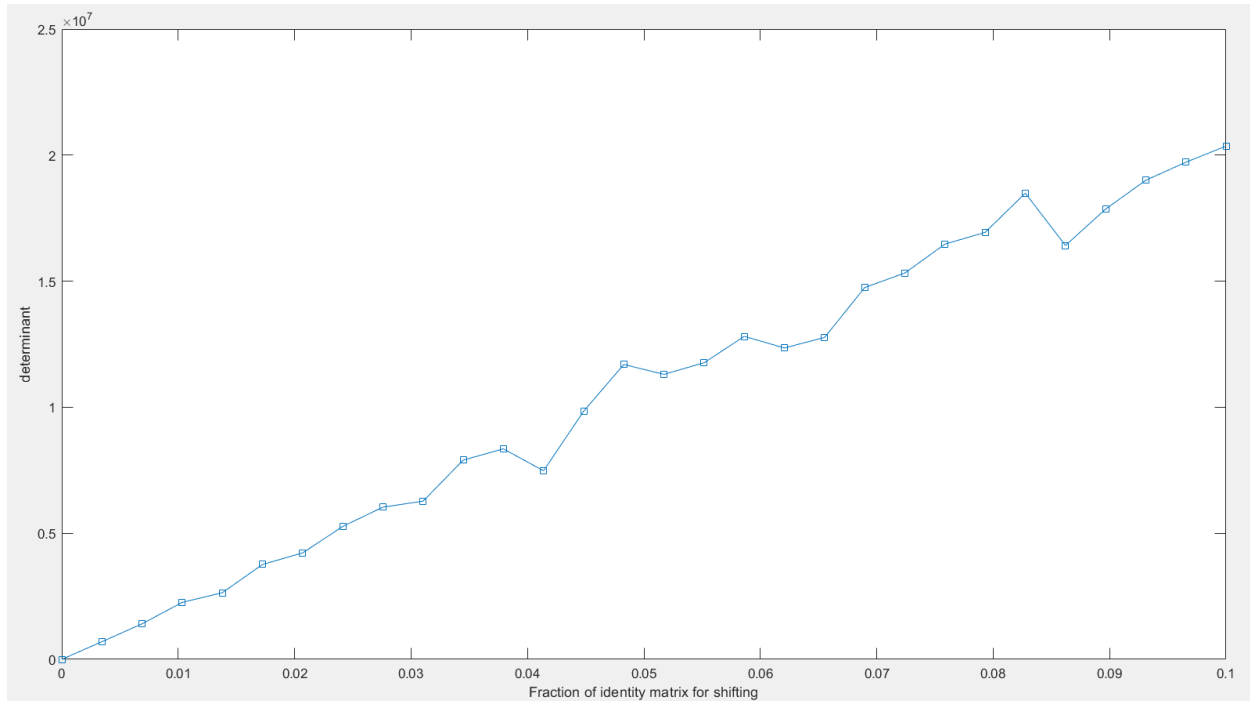


Figure 94 - MATLAB code challenge: determinant of shifted matrices demonstration

101. CODE CHALLENGE: DETERMINANT OF MATRIX PRODUCT

The determinant of a product equals the product of the determinants.

$$\det \mathbf{AB} = (\det \mathbf{A}) (\det \mathbf{B})$$

But when matrix size gets above 15, the computer calculations are not accurate.

MATLAB

Code: Lesson_99_Code_Challenge_Determinant_Product.m

Keywords: randn, det, zeros, plot 's-', set gca 'ylim'

```
%% Lesson_99_Code_Challenge_Determinant_Product

% illustrate that det(AB)-det(A)*det(B) = 0
% 1) for random 3x3 matrices
A = randn(3);
B = randn(3);
AB = A*B;
[ det(A) det(B) det(A)*det(B) det(AB) ]

% 2) in a loop over random matrix sizes up to 40x40
k = 40;
dets = zeros(k,2);

for i=1:k
```

```

A = randn(i);
B = randn(i);
AB = A*B;

dets(i,:) = [ det(A)*det(B) det(AB) ];
end

% illustration of theoretical property breaks down when matrices size > 15
plot(dets(:,1)-dets(:,2), 's-')
set(gca,'ylim',[-1 1])

```

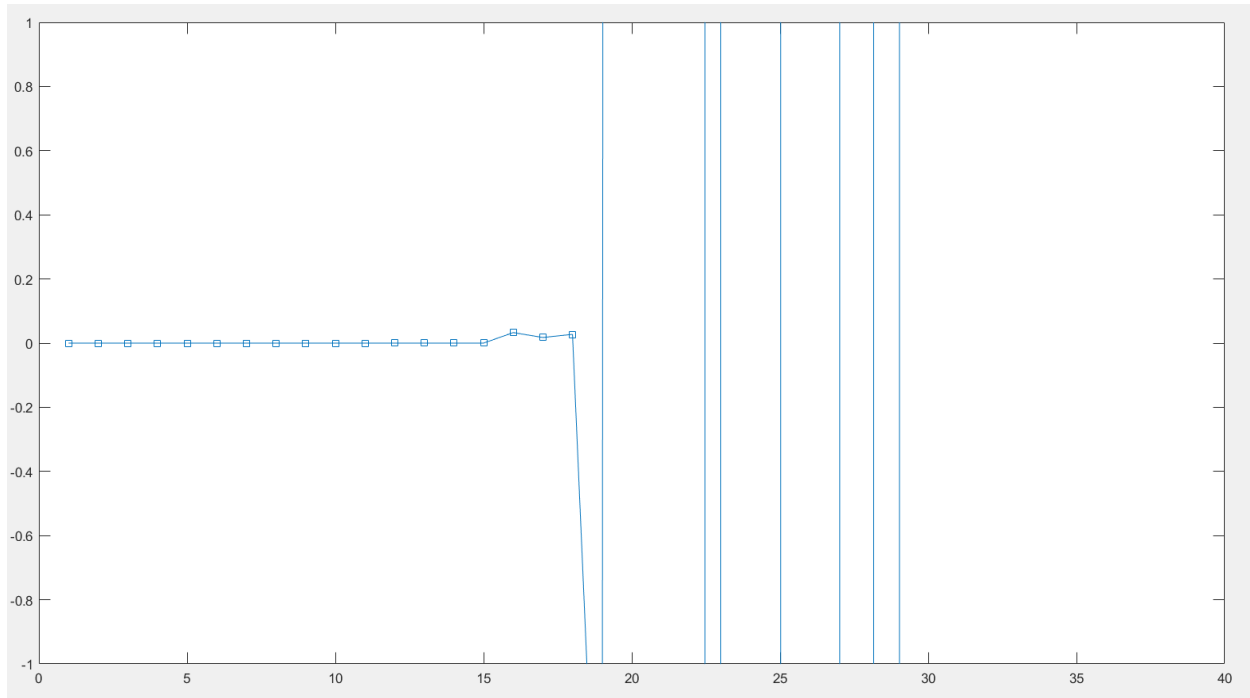


Figure 95 - demonstrate numerical instabilities when computing determinant products

SECTION 10: MATRIX INVERSE

See included exercises with code download:

- linalg_inverse.pdf
- linalg_inverse.m
- linalg_inverse.ipynb

103. MATRIX INVERSE: CONCEPT AND APPLICATIONS

Why a matrix inverse is necessary

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{I}\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Matrix inverse is side-dependent

All matrices in an inverse must be invertible

A matrix is invertible if it is

- Square
- Full rank

Avoid using matrix inverse whenever possible (see code below for demonstration of computation problem)

MATLAB

Code: Lesson_101_Matrix_Inverse.m

Keywords: randn, eps, inv

```
%% Lesson_101_Matrix_Inverse.m
% Create matrix and add noise
A = [ 1 3 9;
      3 6 27;
      4 9 36 ] + randn(3)*eps;

% Try to compute its inverse
Ainv = inv(A)
A*Ainv
```

>> Lesson_100_Matrix_Inverse

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 2.220446e-17.

> In Lesson_100_Matrix_Inverse (line 8)

Ainv =

1.0e+14 *

```
5.6295  5.6295 -5.6295
0.0000 -0.0000  0.0000
-0.6255 -0.6255  0.6255
```

ans =

```
1.0000  0.0625 -0.1250
0  1.2500 -0.2500
0  0.2500  0.5000
```

MATLAB

Code: linalg_inverse.m

Lines: 1 to 32

Keywords: randn, inv, figure, subplot, imagesc, title, axis

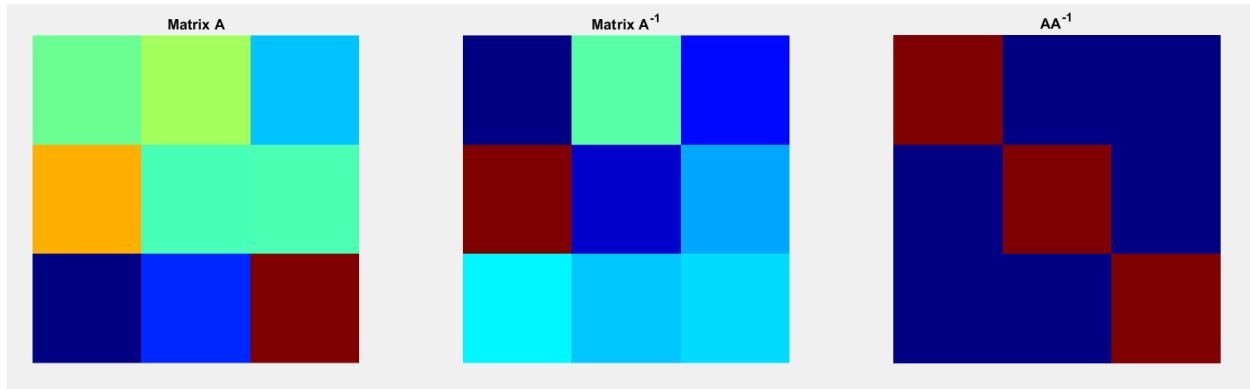


Figure 96 - Use imagesc to demonstrate matrix times its inverse is the identity matrix

105. COMPUTING THE 2X2 MATRIX INVERSE

Inverse is not defined for singular matrices

Check whether a 2x2 matrix has an inverse

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \text{Determinant:} \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

BiophysicsLab.com

Four-step plan

Step 1: Swap diagonal elements

Step 2: Invert off-diagonal signs

Step 3: Divide by determinant to get matrix inverse

Step 4: Verify identity matrix by multiplying original matrix by the inverse

2x2 matrix inverse: the four-step plan

$$\begin{array}{c}
 \begin{pmatrix} a & b \\ c & d \end{pmatrix} \xrightarrow{\text{Step 1}} \begin{pmatrix} d & b \\ c & a \end{pmatrix} \xrightarrow{\text{Step 2}} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \xrightarrow{\text{Step 3}} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \frac{1}{ad - bc} \\
 \\
 \underbrace{\begin{pmatrix} a & b \\ c & d \end{pmatrix}}_{\text{Original Matrix}} \underbrace{\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \frac{1}{ad - bc}}_{\text{Matrix Inverse}} = \begin{pmatrix} ad-bc & -ab+ab \\ cd-cd & -cb+da \end{pmatrix} \frac{1}{ad - bc} \\
 \\
 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}
 \end{array}$$

BiophysicsLab.com

Figure 97 - Four-step plan to compute matrix inverse

106. THE MCA ALGORITHM TO COMPUTE THE MATRIX INVERSE

The MCA algorithm

M: The minors matrix: a matrix of determinants.

C: The cofactors matrix: the minors matrix Hadamard-multiplied by a grid of alternating pluses and minuses.

A⁻¹: The adjugate matrix: the transpose of the cofactors matrix, divided by the determinant.

The minors matrix is the same size as the original matrix. A minor of the matrix element is evaluated by taking the determinant of a submatrix created by deleting the elements in the same row and column as that element. A minor of the element a_{ij} is denoted as M_{ij} .

The MCA algorithm: The *minors matrix*

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \mathbf{M} = \begin{pmatrix} \begin{vmatrix} f & g & h \\ j & k & l \\ n & o & p \end{vmatrix} & \begin{vmatrix} e & & g & h \\ i & & k & l \\ m & & o & p \end{vmatrix} & \begin{vmatrix} e & f & & h \\ i & j & & l \\ m & n & & p \end{vmatrix} & \begin{vmatrix} e & f & g & \\ i & j & k & \\ m & n & o & \end{vmatrix} \\ \begin{vmatrix} & b & c & d \\ & j & k & l \\ & n & o & p \end{vmatrix} & \begin{vmatrix} a & & c & d \\ & f & & \\ i & & k & l \\ m & & o & p \end{vmatrix} & \begin{vmatrix} a & b & & d \\ & & g & \\ i & j & & l \\ m & n & & p \end{vmatrix} & \begin{vmatrix} a & b & c & \\ & & & h \\ i & j & k & \\ m & n & o & \end{vmatrix} \\ \begin{vmatrix} & b & c & d \\ f & g & h \\ & & & \\ i & & & \\ & n & o & p \end{vmatrix} & \begin{vmatrix} a & & c & d \\ e & & g & h \\ & j & & \\ m & & o & p \end{vmatrix} & \begin{vmatrix} a & b & & d \\ e & f & & h \\ & & k & \\ m & n & & p \end{vmatrix} & \begin{vmatrix} a & b & c & \\ e & f & g & \\ & & & l \\ m & n & o & \end{vmatrix} \\ \begin{vmatrix} & b & c & d \\ f & g & h \\ j & k & l \\ & & & \\ m & & & \end{vmatrix} & \begin{vmatrix} a & & c & d \\ e & & g & h \\ i & & k & l \\ & n & & \\ & & o & p \end{vmatrix} & \begin{vmatrix} a & b & & d \\ e & f & & h \\ i & j & & l \\ & & o & \\ & & & p \end{vmatrix} & \begin{vmatrix} a & b & c & \\ e & f & g & \\ i & j & k & \\ & & & p \end{vmatrix} \end{pmatrix}$$

BiophysicsLab.com

Figure 98 - The minors matrix: a matrix of determinants

The minors matrix is the same size as the original matrix. It is a determinate of submatrices.

The MCA algorithm: The *cofactors matrix*
 +/- checkerboard

$$\mathbf{H} = \begin{pmatrix} + & - & + & - \\ - & + & - & + \\ + & - & + & - \\ - & + & - & + \end{pmatrix}$$

$$h_{i,j} = -1^{i+j}$$

BiophysicsLab.com

Figure 99 - The MCA algorithm: The cofactors matrix +/- checkerboard

H is an alternating positive / negative signs matrix starting from the upper left

The MCA algorithm: The *cofactors matrix*

$$\mathbf{C} = \mathbf{M} \odot \mathbf{H}$$

Minors +/- Checkerboard

Hadamard
Multiplication

BiophysicsLab.com

Figure 100 - The MCA algorithm: The cofactors matrix

The cofactors matrix is the minors matrix with alternating sign elements.

The MCA algorithm: The *adjugate matrix*

$$\mathbf{A}^{-1} = \mathbf{C}^T \frac{1}{\text{Det}(\mathbf{A})}$$

BiophysicsLab.com

Figure 101 - The MCA algorithm: The adjugate matrix

The inverse of the original matrix is the transpose of the cofactors matrix multiplied by the determinant of the original matrix.

The MCA algorithm: Example

$$\begin{aligned}
 \mathbf{B} &= \begin{pmatrix} 1 & 3 & 0 \\ 0 & -2 & 8 \\ 7 & 2 & -6 \end{pmatrix} & \text{Det}(\mathbf{B}) &= \begin{vmatrix} 1 & 3 & 0 \\ 0 & -2 & 8 \\ 7 & 2 & -6 \end{vmatrix} \\
 & & &= 12 + 168 + 0 - 0 - 0 - 16 = 164 \\
 & & &\neq 0 \\
 \mathbf{M}_B &= \begin{pmatrix} -4 & -56 & 14 \\ -18 & -6 & -19 \\ 24 & 8 & -2 \end{pmatrix} \\
 \mathbf{H}_B &= \begin{pmatrix} + & - & + \\ - & + & - \\ + & - & + \end{pmatrix} \\
 \mathbf{C}_B &= \begin{pmatrix} -4 & 56 & 14 \\ 18 & -6 & 19 \\ 24 & -8 & -2 \end{pmatrix} \\
 \mathbf{B}^{-1} &= \begin{pmatrix} -4 & 18 & 24 \\ 56 & -6 & -8 \\ 14 & 19 & -2 \end{pmatrix} \frac{1}{164}
 \end{aligned}$$

BiophysicsLab.com

Figure 102 - MCA algorithm example: matrix inverse

107. CODE CHALLENGE: IMPLEMENT THE MCA ALGORITHM

MATLAB

File: Lesson_107_New_Code_Implement_MCA_Algorithmx

Keywords: deal, Boolean addressing, randn, for, det, hadamard multiplication, imagesc, inv

```

% MCA algorithm in code

% compute the inverse using MATLAB inv(). Compare to MCA

m = 4;
A = randn(m);

% initialize (needed when m adjusted to smaller value!)
[minorors,H] = deal( zeros(m) );
    
```

```

% M: minors
for i=1:m
    for j=1:m
        % define the rows and columns for the submatrix
        rows = true(1,m);
        rows(i) = false;

        cols = true(1,m);
        cols(j) = false;

        % compute the minors matrix
        minors(i,j) = det( A(rows,cols) );

        % compute H matrix
        H(i,j) = (-1)^(i+j);
    end
end

% cofactors
C = minors .* H;

% compute Adjugate
Ainv = C'/det(A);

% some code tests
A*Ainv % should produce the identity matrix
imagesc(A*Ainv) % use image scale function to visualize diagonal matrix

Ainv2 = inv(A);
Ainv-Ainv2 % should produce a zeros matrix

```

Results:

```
>> Lesson_107_New_Code_Implement_MCA_Algorithmx
```

ans =

```

1.0000  0.0000 -0.0000   0
    0  1.0000    0    0
    0 -0.0000  1.0000  0.0000
    0  0.0000    0  1.0000

```

ans =

```
1.0e-15 *
```

```

0  0.0555 -0.0555  0.1110
0.0555  0.1110 -0.0278  0.1110
0    0  0.0069  0.1110
-0.2220 -0.2220  0    0

```

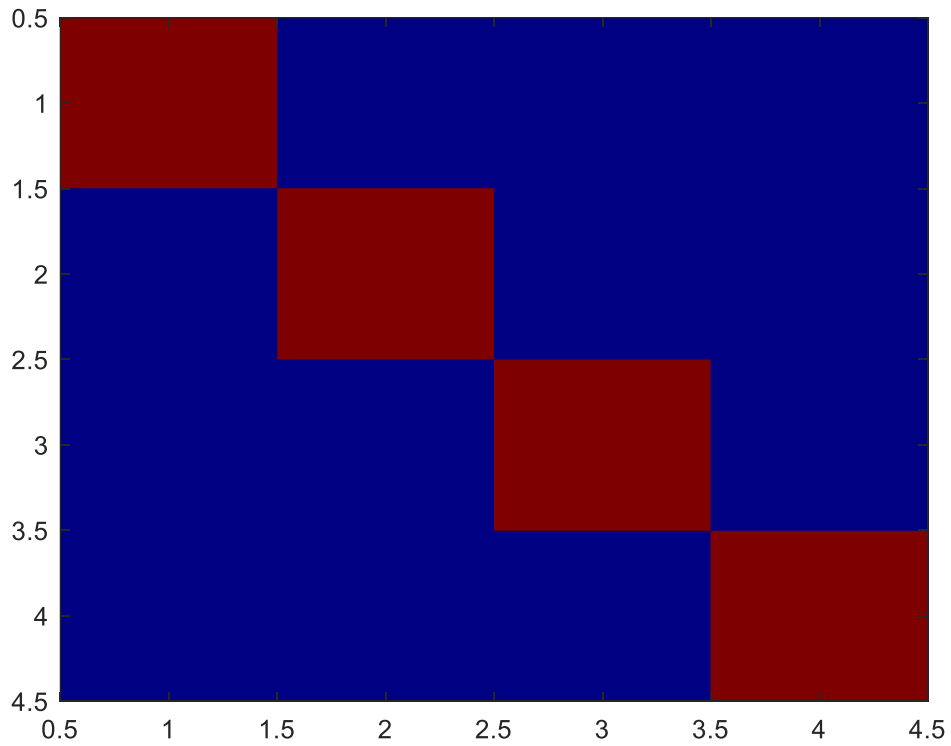


Figure 103 - imagesc of resulting identity matrix

108. COMPUTING THE INVERSE VIA ROW REDUCTION

See prior lecture on Row Reduced Echelon Form (rref): [Lesson 90](#)

Computing inverse via row reduction

$$\text{rref}\left(\left[\begin{array}{c|c} \mathbf{A} & \mathbf{I} \end{array}\right]\right) \rightarrow \left[\begin{array}{c|c} \mathbf{I} & \mathbf{A}^{-1} \end{array}\right]$$

BiophysicsLab.com

Figure 104 - Computing inverse via row reduction

Computing inverse via row reduction: example 1

$$\begin{array}{lcl}
 \left(\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 1 & 3 & 4 & 0 & 1 & 0 \\ 1 & 2 & 5 & 0 & 0 & 1 \end{array}\right) & \begin{array}{l} R_2 = R_2 - R_1 \\ R_3 = R_3 - R_1 \end{array} \Rightarrow & \left(\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 2 & -1 & 0 & 1 \end{array}\right) \\
 R_3 = R_3/2 \Rightarrow & & \left(\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -.5 & 0 & .5 \end{array}\right) & R_2 = R_2 - R_3 \Rightarrow & \left(\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & -.5 & 1 & -.5 \\ 0 & 0 & 1 & -.5 & 0 & .5 \end{array}\right) \\
 R_1 = R_1 - 2R_2 \Rightarrow & & \left(\begin{array}{ccc|ccc} 1 & 2 & 3 & 2 & -2 & 1 \\ 0 & 1 & 1 & -.5 & 1 & -.5 \\ 0 & 0 & 1 & -.5 & 0 & .5 \end{array}\right) & R_1 = R_1 - 3R_3 \Rightarrow & \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 3.5 & -2 & -.5 \\ 0 & 1 & 0 & -.5 & 1 & -.5 \\ 0 & 0 & 1 & -.5 & 0 & .5 \end{array}\right)
 \end{array}$$

BiophysicsLab.com

Figure 105 Computing inverse via row reduction: example 1

MATLAB

Code: linalg_inverse.m

Lines: 33 to 80

Keywords: round, randn, eye, size, rref, inv, figure, clf, subplot, imagesc, title, set, title

```
%%  
% COURSE: Linear algebra: theory and implementation  
% SECTION: Matrix inverse  
% VIDEO: Computing the inverse via row reduction  
% Instructor: sincxpress.com  
%  
%%  
  
% matrix size  
m = 4;  
  
% random integers matrix  
A = round( 10*randn(m) );  
  
% augment A and identity  
Aaug = [ A eye(m) ];  
size(Aaug)  
  
% rref  
Asol = rref(Aaug);  
  
Ainvrref = Asol(:,m+1:end);  
Ainv = inv(A);  
  
% show the augmented matrices  
figure(2), clf  
subplot(211), imagesc(Aaug)  
title('A|I'), axis off  
set(gca,'clim',[-1 1]*max(abs(Aaug(:)))*.5)  
  
subplot(212), imagesc(Asol)  
title('I|A^{-1}'), axis off  
set(gca,'clim',[-1 1]*max(abs(Asol(:)))*.25)  
  
% show the square matrices  
figure(3), clf  
subplot(131), imagesc(A)  
title('Matrix A'), axis square, axis off  
  
subplot(132), imagesc(Ainvrref)  
title('Matrix A^{-1} from rref'), axis square, axis off  
  
subplot(133), imagesc(Ainv)  
title('A^{-1} from inv() function'), axis square, axis off
```

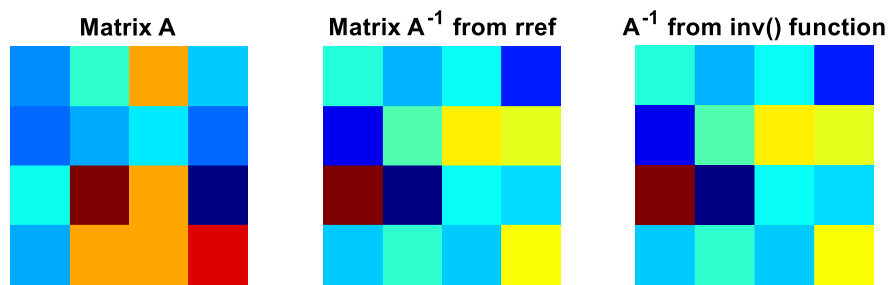


Figure 106 - Visualize square matrix and its inverse using rref and inv functions

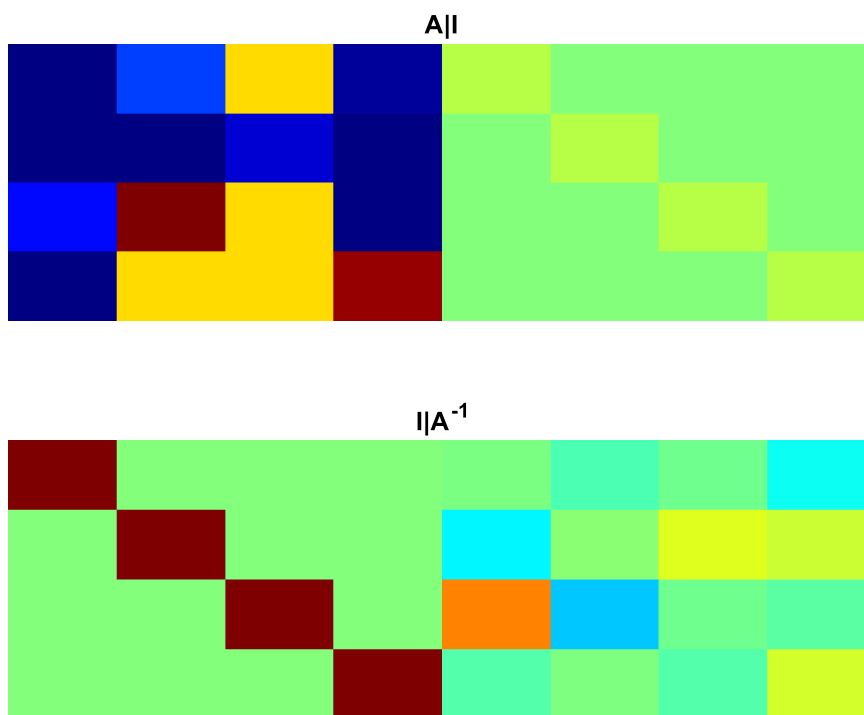


Figure 107 - Visualize augmented matrix with inverse

109. CODE CHALLENGE: INVERSE OF A DIAGONAL MATRIX

The inverse of a matrix is the inverse of each element within the matrix.

MATLAB

Code: Lesson_109_New_Code_challenge_inverse_of_diagonal.m

Keywords: diag, inv

```

% Diagonal matrices, and their inverses

% Create some diagonal matrices, start with 2x2 integers, work up to larger
% matrices
% Compute their inverses (condition on the diagonal matrix for
% invertibility)
% ... think ...

A = [ 2 0; 0 3 ];
A = diag(1:5);

Adiag = diag(A);
Idiag = diag( inv(A) );

% Conditions on invertibility
% Square
% Full rank (maximal number of linearly independent columns,
% ie columns form a linearly independent set)

```

Output:

```

>> Lesson_109_New_Code_challenge_inverse_of_diagonal
>> A

```

A =

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

```

>> Adiag

```

Adiag =

1
2
3
4
5

```

>> Idiag

```

Idiag =

1.0000

0.5000
0.3333
0.2500
0.2000

110. LEFT INVERSE AND RIGHT INVERSE

Rectangular matrices can have an inverse

Condition for inverting rectangular matrices (maximum possible rank)

A has a left inverse if it is **full column-rank**

A has to be rectangular or tall, wide matrix does not have a left inverse

Left inverse for tall matrices

$$A = \begin{bmatrix} \end{bmatrix}_{m \times n, m > n, \text{rank} = n} \quad (A^T A)^{-1} \underbrace{(A^T A)}_{n \times n} = I$$

$$\underbrace{(A^T A)^{-1} A^T}_{\text{"Left inverse"}} A = I$$

BiophysicsLab.com

Figure 108 - Left inverse for tall matrices

A has a right inverse if it is **full row-rank**

111. ONE-SIDED INVERSES IN CODE

MATLAB

Code: linalg_inverse.m

Lines: 81 to 144

Keywords: randn, matrix complex conjugate $'$, disp, num2str, inv, figure, subplot, imagesc, axis image, axis off, title

```
%%  
% COURSE: Linear algebra: theory and implementation  
% SECTION: Matrix inverse  
% VIDEO: Left inverse and right inverse  
% Instructor: sincxpress.com  
%  
%%  
  
% m>n for left inverse,  
% m<n for right inverse  
m = 6;  
n = 3;  
  
% create matrices  
A = randn(m,n);  
AtA = A'*A;  
AAAt = A*A';  
  
% inspect ranks  
disp(['Rank of A^TA: ' num2str(rank(AtA)) ])   
disp(['Rank of AA^T: ' num2str(rank(AAAt)) ])   
  
% left inverse  
Aleft = inv(AtA)*A';  
  
% right inverse  
Aright = A'*inv(AAAt);  
  
% now test!  
I_left = Aleft * A;  
I_right = A * Aright;  
  
% and then test using the inverse function  
AtA_inv = inv(AtA);  
I_AtA = AtA_inv * AtA;  
  
AAAt_inv = inv(AAAt);  
I_AAAt = AAAt_inv * AAAt;  
  
% show images  
figure(4), clf  
subplot(331), imagesc(A), axis image, axis off  
title(['A, r=' num2str(rank(A)) ])   
  
subplot(332), imagesc(AtA), axis image, axis off  
title(['A^TA, r=' num2str(rank(AtA)) ])   
  
subplot(333), imagesc(AAAt), axis image, axis off
```

```

title(['AA^T, r=' num2str(rank(AA^T)) ])

subplot(335), imagesc(Aleft), axis image, axis off
title('Left inverse: (A^T A)^{-1} A^T')

subplot(336), imagesc(Aright), axis image, axis off
title('Right inverse: A^T (A A^T)^{-1}')

subplot(338), imagesc(l_left), axis image, axis off
title(['(A^T A)^{-1} A^T A'])

subplot(339), imagesc(l_right), axis image, axis off
title(['A [A^T (A A^T)^{-1}]'])

```

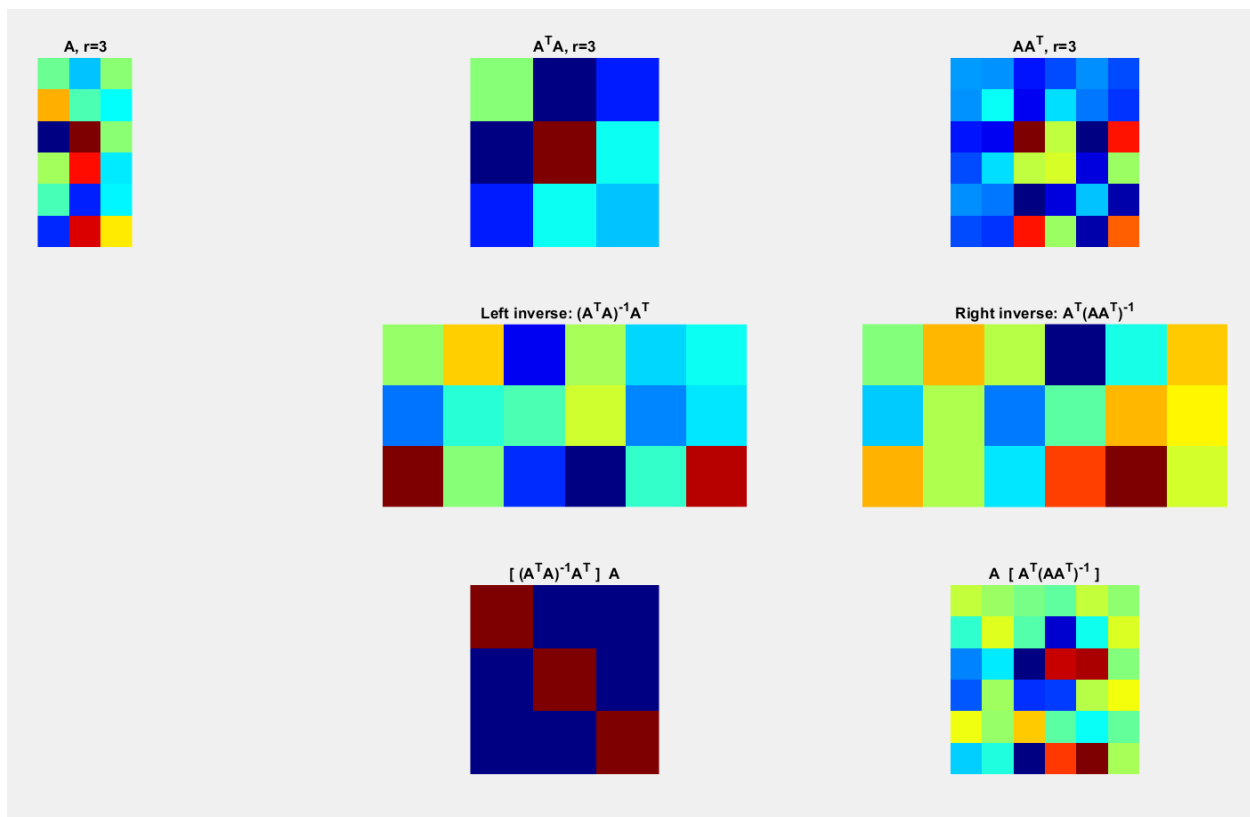


Figure 109 - Demonstration of left inverse

112. PROOF: THE INVERSE IS UNIQUE

For any invertible matrix, there is exactly one unique inverse

Assumptions:

1. **A** is invertible
2. **B** is an inverse of **A**
3. **C** is a distinct (separate) inverse of **A**

Proof 1:

$$\left. \begin{array}{l} \mathbf{A}^{-1}\mathbf{A}\mathbf{B} = \mathbf{A}^{-1}\mathbf{I} \rightarrow \mathbf{B} \\ \mathbf{A}^{-1}\mathbf{A}\mathbf{C} = \mathbf{A}^{-1}\mathbf{I} \rightarrow \mathbf{C} \end{array} \right\} \mathbf{B} = \mathbf{C} \quad \begin{array}{l} = \mathbf{A}^{-1} \\ = \mathbf{A}^{-1} \end{array}$$

Proof 2:

$$\mathbf{C} = \mathbf{C}\mathbf{I} = \mathbf{C}\mathbf{A}\mathbf{B} = \mathbf{I}\mathbf{B} = \mathbf{B}$$

113. PSEUDO-INVERSE, PART 1

Moore-Penrose pseudoinverse is based on singular value decomposition (SVD).

Pseudoinverse is written as \mathbf{A}^* or \mathbf{A}^\dagger , not \mathbf{A}^{-1}

Where \mathbf{A} is a rank deficient square matrix

Pseudoinverse, part 1 - Example

$$\mathbf{A} \quad \mathbf{A}^* \quad \mathbf{A}\mathbf{A}^*$$

$$\begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 0 \\ 3 & 3 & 3 \end{pmatrix} \quad 27 \begin{pmatrix} -5 & 8.5 & 3.5 \\ 1 & 1 & 2 \\ 7 & -6.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 2 \\ 0 & 5 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

BiophysicsLab.com

Figure 110 - Pseudoinverse, part 1 - Example

MATLAB

Code: linalg_inverse.m

Lines: 145 to 195

Keywords: pinv, subplot, imagesc, axis image, axis off, title

```
%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Matrix inverse
% VIDEO: The pseudoinverse, part 1
% Instructor: sincxpress.com
%
%%

%% run the code from the previous video before running this cell!

% pseudoinverse of rectangular matrix A
```



```

pseudoinvA = pinv(A);

subplot(334), imagesc(pseudoinvA), axis image, axis off
title('MP Pseudoinverse of A')

subplot(337), imagesc(pseudoinvA*A), axis image, axis off
title('A^*A')

%%

% create random matrix
n = 50;
A = randn(n);

% make rank deficient by repeating a column
A(:,end) = A(:,end-1);

% rank of A!
rank(A)

% compute the pseudoinverse
Ai = pinv(A);

% and show the matrices
figure(5), clf
subplot(221), imagesc(A), axis square, axis off
title('A'), set(gca,'fontsize',20)

subplot(222), imagesc(Ai), axis square, axis off
title('A^*'), set(gca,'fontsize',20)

subplot(223), imagesc(Ai*A), axis square, axis off
title('A^*A'), set(gca,'fontsize',20)

subplot(224), imagesc(A*Ai), axis square, axis off
title('AA^*'), set(gca,'fontsize',20)

```

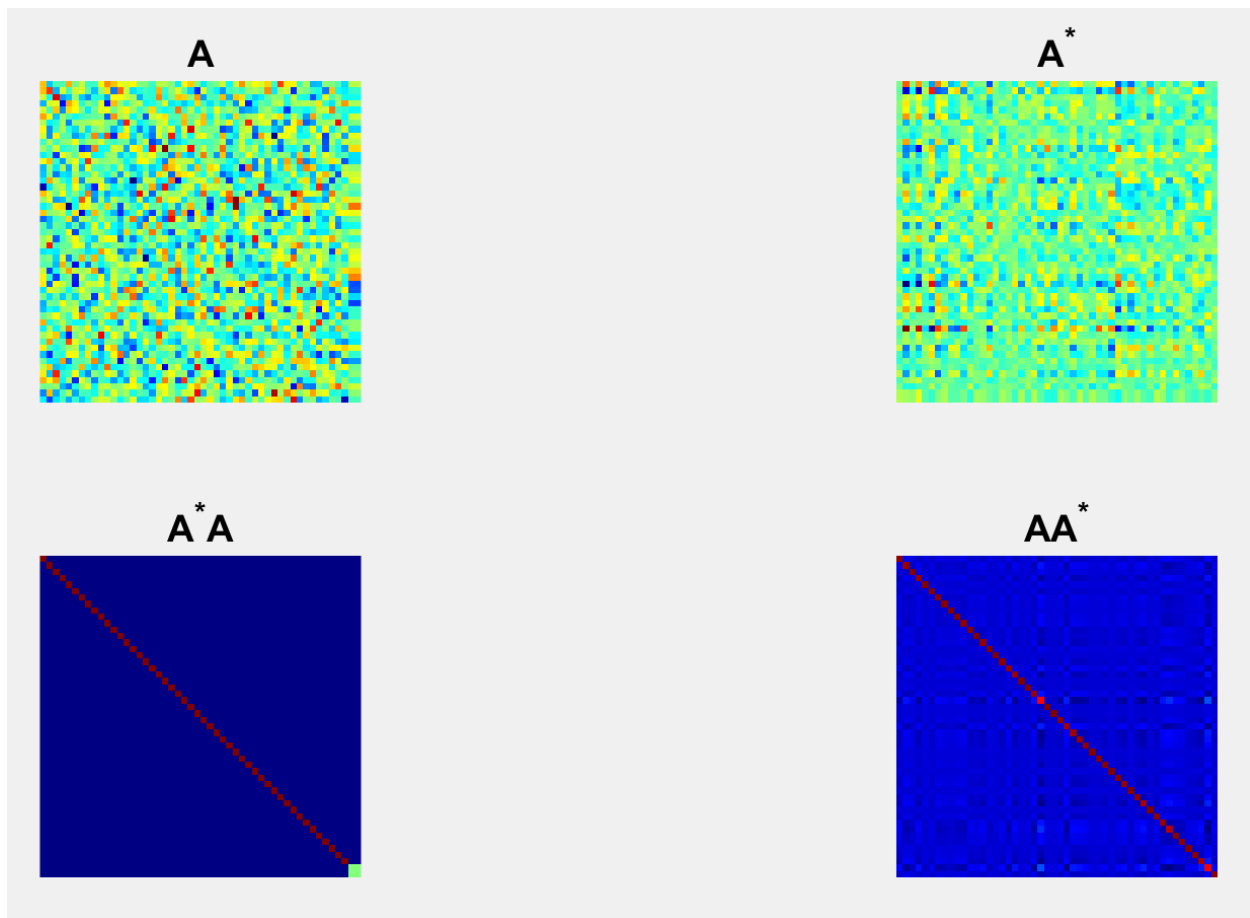


Figure 111 - imagesc demonstration of pseudoinverse

114. Code Challenge

MATLAB

Code: Lesson_114_New_Code_Challenge_Pseudo_vs_Real_Inverse.m

Keywords: randn, inv, pinv

```
% Pseudoinverse is the same as the "real" inverse for an invertible matrix?

m = 5;
A = randn(m);

AinvF = inv(A);
AinvP = pinv(A);

A*AinvF
A*AinvP

% Yes, pseudoinverse is the same as the "real" inverse for this invertible
% matrix, but not a formal proof.
```

Results:

```
>> Lesson_114_New_Code_Challenge_Pseudo_vs_Real_Inverse
```

```
ans =
```

1.0000	-0.0000	0.0000	-0.0000	-0.0000
-0.0000	1.0000	-0.0000	0.0000	0.0000
0.0000	0.0000	1.0000	-0.0000	0.0000
-0.0000	-0.0000	-0.0000	1.0000	0.0000
0.0000	0.0000	0.0000	0.0000	1.0000

ans =

1.0000	-0.0000	0.0000	0.0000	-0.0000
-0.0000	1.0000	-0.0000	0.0000	0.0000
0.0000	-0.0000	1.0000	-0.0000	-0.0000
-0.0000	0.0000	0.0000	1.0000	0.0000
-0.0000	-0.0000	0.0000	0.0000	1.0000

115. WHY SHOULD YOU AVOID THE INVERSE?

Matrix inverse can be unstable and inaccurate.

Failure scenarios will be demonstrated in upcoming lectures: 155 and 170

SECTION 11: PROJECTIONS AND ORTHOGONALIZATION

See included exercises with code download:

- linalg_projorth1.pdf
- linalg_projorth.m
- linalg_projorth.ipynb

108. PROJECTIONS IN \mathbb{R}^2

Old lecture number was 117

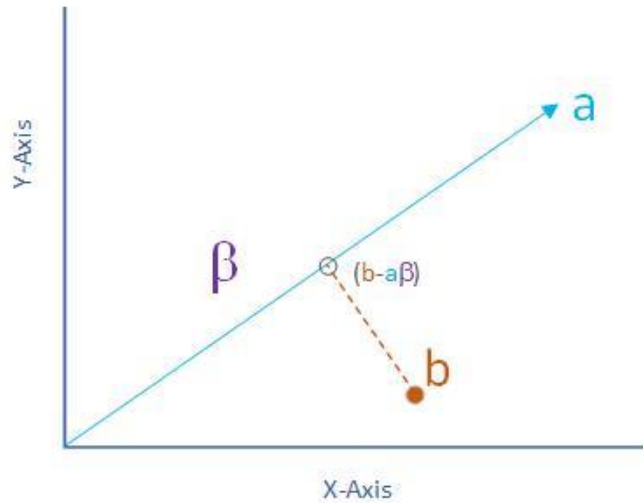
Know the formula for projecting a point onto a line

See a common format for formula: mapping over magnitude

Mapping over magnitude is used for:

- Correlation
- Various normalizations
- Spectral coherence
- Statistics

Projections in \mathbb{R}^2



BiophysicsLab.com

Figure 112 - Projections in \mathbb{R}^2 – Projecting point b onto line a

$$a^T(b - a\beta) = 0$$

$$a^T b - a^T a \beta = 0$$

$$a^T a \beta = a^T b$$

$$\text{"projection"} \quad \beta = \frac{a^T b}{a^T a}$$

$$\text{proj}_a b = \beta a$$

MATLAB

Code: linalg_projorth.m

Lines: 1 to 33

Keywords: matrix complex conjugate ‘, figure, plot, hold on, legend, axis square, get(gca,’xlim’), norm

```
%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Projections and orthogonalization
% VIDEO: Projections in R^2
% Instructor: sincxpress.com
%
%%

% point b
b = [ 4 1 ]';

% line a
a = [ 2 5 ]';

% beta
beta = (a'*b) / (a'*a);
```

```

% draw!
figure(1), clf
plot(b(1),b(2),'ko','markerfacecolor','m','markersize',20)
hold on
plot([0 a(1)],[0 a(2)],'b','linew',3)

% now plot projection line
plot([b(1) beta*a(1)],[b(2) beta*a(2)],'r--','linew',3)

legend({'b';'a';'b-\beta a'})
axis([-1 1 -1 1]*max([norm(a) norm(b)]))
plot(get(gca,'xlim'),[0 0],'k--')
plot([0 0],get(gca,'xlim'),'k--')
axis square

```

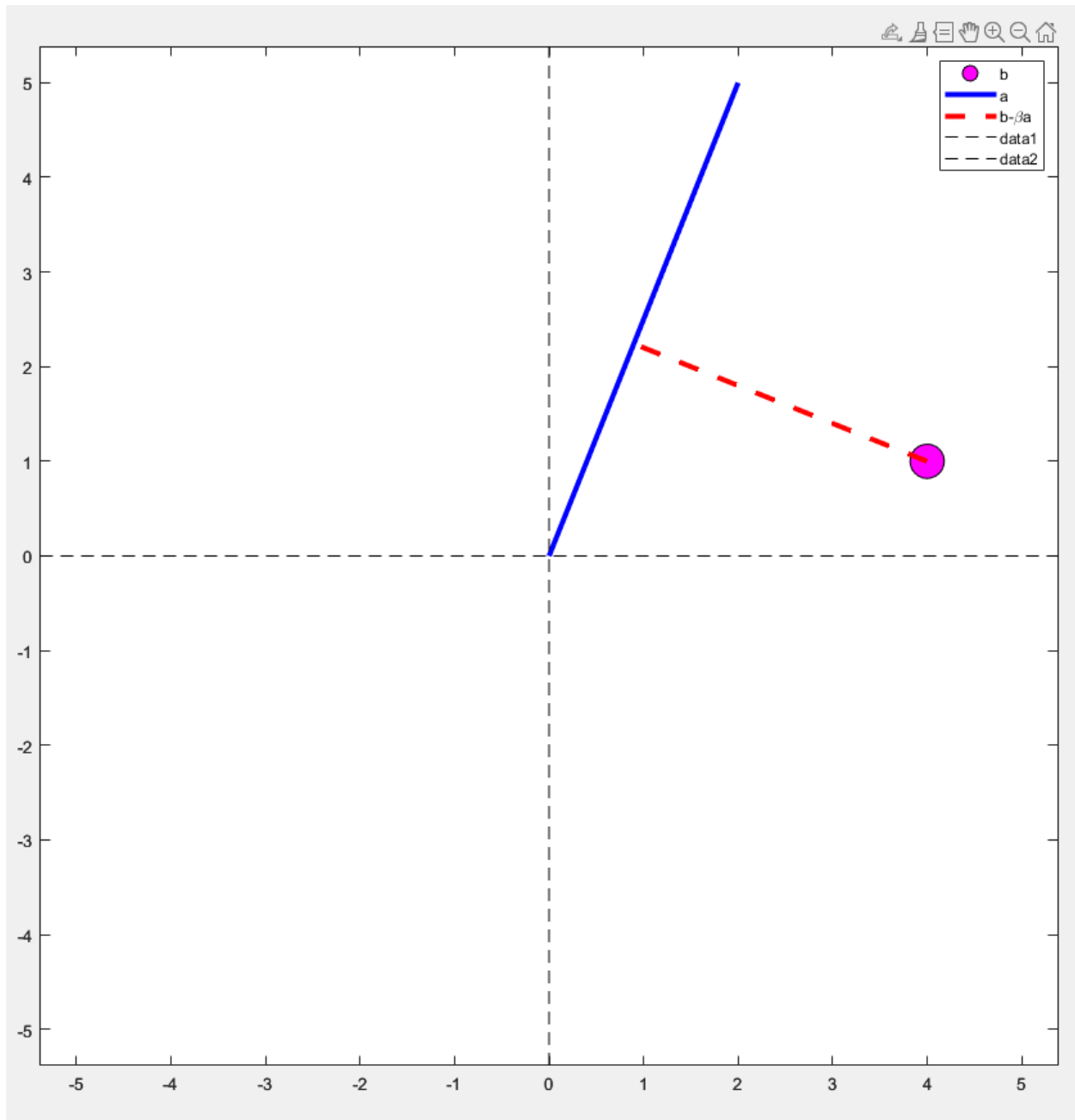


Figure 113 -MATLAB demonstration of projection in R^2

109. PROJECTIONS IN R^N

See how the projection formula is extended from vectors to matrices.

Know the condition on A for the projection formula to be valid:

$A^T A$ must be:

- invertible,
- square (or tall rectangular, i.e., full column rank) matrix,

- full rank

Projections in \mathbb{R}^N

Assume A is Square Matrix

$$\begin{aligned}
 A^T(b - Ax) &= 0 \\
 A^Tb - A^TAx &= 0 \\
 A^TAx &= A^Tb
 \end{aligned}
 \qquad
 \begin{aligned}
 (A^TA)^{-1}(A^TA)x &= (A^TA)^{-1}A^Tb \\
 \underbrace{(A^TA)^{-1}(A^TA)}_{\text{Identity Matrix}} x &= (A^TA)^{-1}A^Tb \\
 x &= (A^TA)^{-1}A^Tb
 \end{aligned}$$

BiophysicsLab.com

Figure 114 - Projection in \mathbb{R}^N

MATLAB

Code: linalg_projorth.m

Lines: 34 to 94

Keywords: randn, inv vs A\b, figure, clf, ezmesh, set, xlabel, ylabel, zlabel, axis, grid, fplot3, plot3, title, legend

```

%%
% COURSE: Linear algebra: theory and implementation
% SECTION: Projections and orthogonalization
% VIDEO: Projections in  $\mathbb{R}^N$ 
% Instructor: sincxpress.com
%
%%

% goal here is to solve Ax=b for x

% sizes
m = 16;
n = 10;

% vector b
b = randn(m,1);

% matrix A
A = randn(m,n);

% solution using explicit inverse

```

```

x1 = inv(A'*A) * (A'*b);

% preferred solution
x2 = (A'*A) \ (A'*b);

% also possible (version dependent)
x3 = A\b;

%% geometric perspective in R^3

m = 3;
n = 2;

% vector b
b = randn(m,1);

% matrix A
A = randn(m,n);

% solution
x = (A'*A) \ (A'*b);
Ax = A*x;

% draw the plane spanned by A
figure(2), clf
h = ezmesh( @(s,t)A(1,1)*s+A(1,2)*t , @(s,t)A(2,1)*s+A(2,2)*t , @(s,t)A(3,1)*s+A(3,2)*t , [-1 1 -1
1]*norm(x)*2);
set(h,'facecolor','g','cdata',ones(50),'EdgeColor','none')
xlabel('R_1'), ylabel('R_2'), zlabel('R_3')
axis square
grid on, rotate3d on, hold on

h(1) = fplot3( @(t)t*b(1), @(t)t*b(2), @(t)t*b(3) , [-1 1]);
h(2) = fplot3( @(t)t*Ax(1), @(t)t*Ax(2), @(t)t*Ax(3) , [-1 1]);
plot3(Ax(1),Ax(2),Ax(3),'ro','markerfacecolor','r','markersize',12)

set(h,'LineWidth',3)
title('')
legend({'C(A)';'b';'Ax'})

```

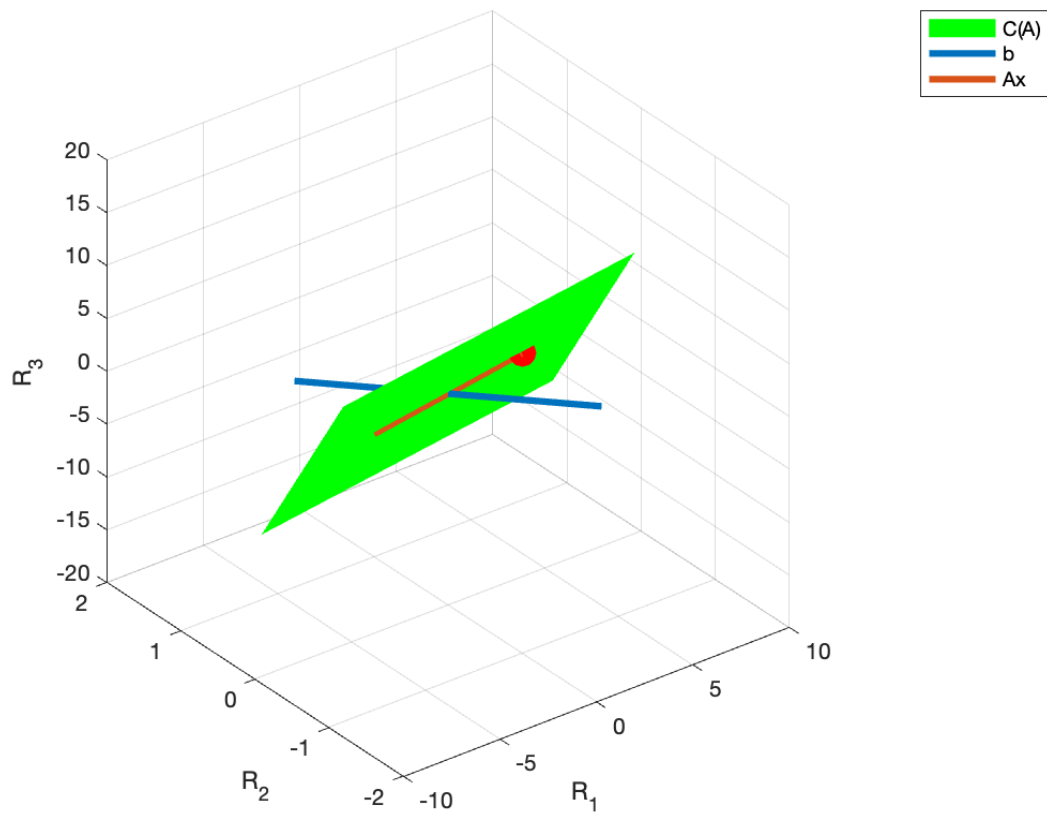



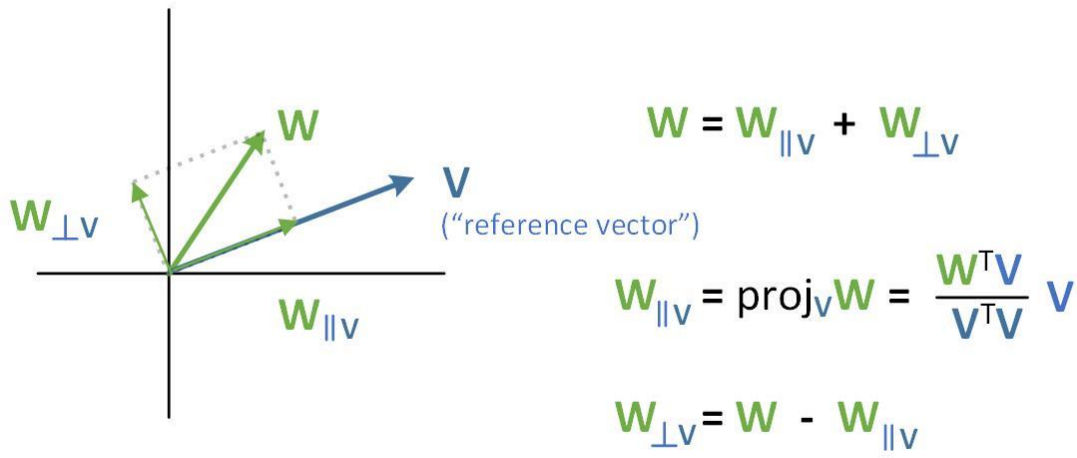
Figure 115 - MATLAB demonstration of projection in R^N

110. ORTHOGONAL AND PARALLEL VECTOR COMPONENTS

Know how to decompose a vector into two orthogonal vectors.

Gain more experience in using the link between geometry and algebra to discover and understand concepts in linear algebra.

Orthogonal and parallel vector components (Separating two components of a vector)



BiophysicsLab.com

Figure 116 - Orthogonal and parallel vector components

Proof that $\mathbf{W}_{\parallel \mathbf{V}}$ is orthogonal to $\mathbf{W}_{\perp \mathbf{V}}$ by verifying that their dot product equals zero.

Where:

$$(\mathbf{W}_{\parallel \mathbf{V}})^T (\mathbf{W}_{\perp \mathbf{V}}) = \underbrace{\left(\frac{\mathbf{W}^T \mathbf{V}}{\mathbf{V}^T \mathbf{V}} \mathbf{V} \right)^T}_{\text{Projection of } \mathbf{W} \text{ onto the line } \mathbf{V}} \cdot \left(\mathbf{W} - \frac{\mathbf{W}^T \mathbf{V}}{\mathbf{V}^T \mathbf{V}} \mathbf{V} \right) = 0$$

Projection of \mathbf{W} onto the line \mathbf{V}

$$\begin{aligned}
 &= \frac{\mathbf{W}^T \mathbf{V}}{\mathbf{V}^T \mathbf{V}} \mathbf{V}^T \mathbf{W} - \frac{\mathbf{W}^T \mathbf{V}}{\mathbf{V}^T \mathbf{V}} \frac{\mathbf{W}^T \mathbf{V}}{\mathbf{V}^T \mathbf{V}} \mathbf{V}^T \mathbf{V} \\
 &= \frac{(\mathbf{W}^T \mathbf{V})^2}{\mathbf{V}^T \mathbf{V}} - \frac{(\mathbf{W}^T \mathbf{V})^2}{\mathbf{V}^T \mathbf{V}} \\
 &= 0
 \end{aligned}$$

Orthogonal and Parallel Vector Components - Example

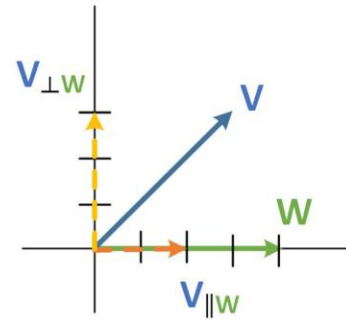
Given \mathbf{W} as a reference vector. Calculate the orthogonal and parallel vector components of \mathbf{V} .

$$\mathbf{W} = \begin{bmatrix} 4 \\ 0 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$\mathbf{V}_{\parallel \mathbf{W}} = \frac{\mathbf{W}^T \mathbf{V}}{\mathbf{W}^T \mathbf{W}} \mathbf{W} = \frac{4 \times 2 + 0 \times 3}{4 \times 4 + 0 \times 0} \begin{bmatrix} 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$\mathbf{V}_{\perp \mathbf{W}} = \mathbf{V} - \mathbf{V}_{\parallel \mathbf{W}} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

Plot the example showing \mathbf{W} , \mathbf{V} , $\mathbf{V}_{\parallel \mathbf{W}}$, and $\mathbf{V}_{\perp \mathbf{W}}$



BiophysicsLab.com

Figure 117 - Orthogonal and parallel vector components - Example

111. CODE CHALLENGE DECOMPOSE VECTOR

Implement the procedures discussed in previous lecture. Decompose one vector relative to another vector into parallel and orthogonal components.

MATLAB

Code: Lesson_111_Code_Challenge_Decompose_vector.m

Keywords: row vs column vector, plot, hold on, legend, axis, grid

```
%% Lesson_111_Code_Challenge_Decompose_vector.m

% vector w, to be decomposed
w = [ 2 3 ]';

% vector v, the reference
v = [ 4 0 ]';

% compute w-parallel-to-v [a dot product]
% notes:
% valid only if w and v are both column vectors.
% if they were both row vectors, the result of beta would be an outer product
beta = (w'*v) / (v'*v);
% alternative methods:
% beta = dot(w,v) / (v'*v);
% beta = sum(w.*v) / (w'*v); where sum uses an element-wise multiplication
w_par_v = beta*v;

% compute w-orthogonal-to-v
w_perp_v = w - w_par_v;
```

```

% confirm results algebraically (sum to w, orthogonal components)
(w_par_v + w_perp_v) - w
w_par_v*w_perp_v

% plot all four vectors!
figure(1), clf, hold on
plot([0 w(1)], [0 w(2)], 'r', 'linewidth', 3)
plot([0 v(1)], [0 v(2)], 'b', 'linewidth', 2)

plot([0 w_par_v(1)], [0 w_par_v(2)], 'r--', 'linewidth', 3)
plot([0 w_perp_v(1)], [0 w_perp_v(2)], 'r:', 'linewidth', 3)

legend({'w'; 'v'; 'w_{||v}'; 'w_{\perp v}'})
axis([-1 1 -1 1]*5)
axis square
grid on

```

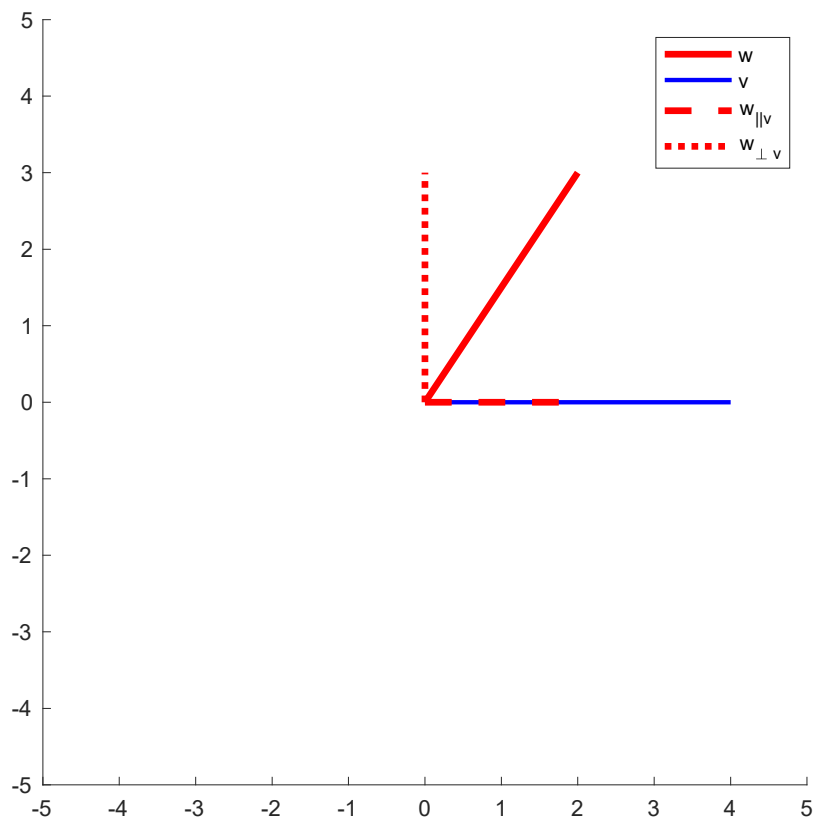


Figure 118 - Lesson 111_Code_Challenge_Decompose_Vector

112. ORTHOGONAL MATRICES

What does it mean for a matrix to be orthogonal:

- Know the definition and key property of an orthogonal matrix.
- See yet another example of how geometric insight helps you learn linear algebra.
- 1. Use the letter **Q** to indicate an orthogonal matrix.
- 2. All columns are pairwise orthogonal.
 - a. Meaning that the dot product between any two columns will be 0.
- 3. Each column has a magnitude = 1.
 - a. Where magnitude is the length or the norm of each column.

GLOSSARY

- Adjugate: In linear algebra, the adjugate or classical adjoint of a square matrix is the transpose of its cofactor matrix.
- Canonical form: "In mathematics and computer science, a canonical, normal, or standard form of a mathematical object is a standard way of presenting that object as a mathematical expression. Often, it is one which provides the simplest representation of an object and which allows it to be identified in a unique way." - https://en.wikipedia.org/wiki/Canonical_form
- Determinant: The determinant of a matrix is a scalar value calculated from the elements of a Square Matrix (matrix with $m = n$). Specific properties of the determinant make them useful for solving the linear system of equations, checking the invertibility of a matrix, finding the area and volume of geometric shapes.
- Echelon: "An echelon is a stepped formation with objects arranged in a diagonal. Birds flying in a V shape create echelons so that they can draft behind each other and conserve energy."
<https://www.vocabulary.com/dictionary/echelon>
- Full rank matrix: A matrix is full row rank when each of the rows of the matrix are linearly independent and full column rank when each of the columns of the matrix are linearly independent. For a square matrix these two concepts are equivalent and we say the matrix is full rank if all rows and columns are linearly independent. A square matrix is full rank if and only if its determinant is nonzero.
 - For a non-square matrix with rows and columns, it will always be the case that either the rows or columns (whichever is larger in number) are linearly dependent. Hence when we say that a non-square matrix is full rank, we mean that the row and column rank are as high as possible, given the shape of the matrix. So if there are more rows than columns ($m > n$), then the matrix is full rank if the matrix is full column rank.
- The Fundamental theorem of Algebra: The fundamental theorem of algebra states that you will have n roots for an n th degree polynomial.
- Hadamard multiplication: The Hadamard product is a binary operation that takes two matrices of the same dimensions and produces another matrix of the same dimension as the operands, where each element i, j is the product of elements i, j of the original two matrices.
- Invertible matrix: A matrix with maximum rank, a.k.a. full rank set
- Orthogonal complement: Let V be a vector space and W be a subspace of V . Then the orthogonal complement of W in V is the set of vectors u such that u is orthogonal to all vectors in W .

- Penultimate: “next to last” or “second to last.”
- Rank-deficient matrix: A matrix with less than maximum rank.
- Singular matrix: see Rank-deficient matrix.
- Subset: Unlike a **subspace** a subset is set of points that doesn’t need to include the origin; doesn’t need to be closed under addition or multiplication; can have boundaries.
- Subspace: A subspace is defined as a set of all vectors that can be created by taking linear combinations of some vector or a set of vectors. Where linear combination means multiplying a vector by a scalar. Also combining two vectors with their associated scalar multipliers also forms a subspace. The origin must be included.
- Symmetric Matrix: A symmetric matrix is a square matrix that satisfies $\mathbf{A}^T = \mathbf{A}$.

REFERENCES

Interactive Linear Algebra by Dan Margalit and Joseph Rabinoff

<https://textbooks.math.gatech.edu/ila/>

Properties of determinants – MIT OpenCourseWare

https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/least-squares-determinants-and-eigenvalues/properties-of-determinants/MIT18_06SCF11_Ses2.5sum.pdf

Determinant of a Matrix | Linear Algebra Using Python

<https://www.codeformech.com/determinant-linear-algebra-using-python/>