MATH2221
Mathematics Laboratory II

Lecture 8:
Advanced Linear Algebra Functions in MATLAB

Gary Choi

March 11, 2025

About Test 1

- Statistics:
  - Full mark = 60
  - Max = 60
  - Mean = 42.5
  - SD = 11.1

Image source: Squid Game

- For enquires, please contact the corresponding graders:
  - Q1: Ms. WEI Yulin (LSB 222B, ylwei@math.cuhk.edu.hk)
  - Q2: Mr. TSANG Hei Tung (Science Center 333B, httsang@math.cuhk.edu.hk)
  - Q3: Mr. HUANG Yanwen (Science Center 333B, ywhuang@math.cuhk.edu.hk)
  - Q4: Mr. JIANG Qinghai (LSB 222B, qhjiang@math.cuhk.edu.hk)
  - Q5: Gary (LSB 204, ptchoi@cuhk.edu.hk)

Reminder: More learning resources and exercises

- *MATLAB: A Practical Introduction to Programming and Problem Solving* by S. Attaway
  - Online access to the 5th edition (2019) is available via CUHK LibrarySearch (https://www.lib.cuhk.edu.hk/en/)

- *MATLAB by Example: Programming Basics* by M. Gdeisat, F. Lilley, Elsevier Science, 2013.
  - Online access to the full text is available via CUHK LibrarySearch (https://www.lib.cuhk.edu.hk/en/)

- *MATLAB Academy* https://matlabacademy.mathworks.com/
  - Many self-paced online courses and exercises are freely available

Reminder: Accessing MATLAB outside the classes

- Our computing lab (LSB 232B)
  - Open 24 hours (except for time slots reserved for lab classes)

- CUHK Library / Pi Chiu Building /Learning Commons computers
  https://www.lib.cuhk.edu.hk/en/use/facilities/computer/
  https://www.itsc.cuhk.edu.hk/all-it/it-facilities/user-areas/software-in-user-areas/

- Download and install on your own computers
  - FREE license for all CUHK students
    https://www.itsc.cuhk.edu.hk/all-it/procurement-support/campus-wide-software/matlab-and-simulink/

What's next?

Lecture 1 – 7: MATLAB Basics

- Scalar/vector/matrix operations
- Writing MATLAB functions
- Relational and logical operators
- if/for/while statements, recursion
- 2D visualization
- 3D visualization

Lecture 8 – 13: Advanced topics

- Advanced linear algebra functions
- File input/output
- Data analysis
- Image/video processing
- Calculus and optimization
- Symbolic computation

New topic: Advanced Linear Algebra functions

- **Recall:**

    - Basic vector/matrix computations: A*B, A^2, A*u, v'*v, ...
    - Entrywise operations: A.*B, A./B, u.^v, …
    - Solving linear system: A\b
    - Explicit matrix inverse: inv(A), A^(-1)
    - Sum and product: sum(v), prod(v)
    - Dot product: dot(u,v)
    - Cross product: cross(u,v)
    - Trace: trace(A)
    - Determinant: det(A)

- What other linear algebra functions are available?

Vector norm

- **Vector norm** of a $n \times 1$ vector v: $\|v\|_p = (\sum_{i=1}^{n} |v_i|^p)^{1/p}$

- MATLAB command: norm(v,p) where p = any positive real scalar, Inf, or -Inf
  - p = 1 equivalent to sum(abs(v))
  - p = 2 (default) equivalent to sum(abs(v).^2)^(1/2)
  - p = positive real scalar equivalent to sum(abs(v).^p)^(1/p)
  - p = Inf equivalent to max(abs(v))
  - p = -Inf equivalent to min(abs(v))

- Example:

```
>> v = [1, 2, 3];
>> norm(v,2)   % i.e. sqrt(1^2 + 2^2 + 3^2)
ans =
    3.7417
```

Matrix norm

- **Matrix norm** of a $m \times n$ matrix A: $\|A\|_p = \sup\{\|Ax\|_p : \|x\|_p \le 1\}$
  - $p = 1 : \|A\|_1 = \max\limits_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$ (maximum absolute column sum)
  - $p = 2 : \|A\|_2 = \sqrt{\lambda_{max}(A^* A)}$ (square root of the largest eigenvalue of $A^* A$, where A^* is the conjugate transpose of A)
    (Remark: In MATLAB, A' gives the conjugate transpose, while A.' gives the nonconjugate transpose)
  - $p = \infty : \|A\|_\infty = \max\limits_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$ (maximum absolute row sum)

- MATLAB command: norm(A,p)
  - p = 1 equivalent to max(sum(abs(X),1))
  - p = 2 (default) equivalent to max(svd(X))
  - p = Inf equivalent to max(sum(abs(X),2))

- Example:

```
>> A = [-3, 5, 7; 2, 6, 4; 0, 2, 8];
>> norm(A,1)   % i.e. max{|-3|+2+0, 5+6+2, 7+4+8}
```

```
ans =
   19
```

Reduced row echelon form (RREF)

- **Reduced row echelon form**:
  - It is in row echelon form.
  - The leading entry in each nonzero row is 1.
  - Each column containing a leading 1 has zeros in all its other entries.

e.g. $\begin{pmatrix} 1 & 0 & a & 0 & c \\ 0 & 1 & b & 0 & d \\ 0 & 0 & 0 & 1 & e \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

- MATLAB command: R = rref(A)

- Example: Solve $\begin{cases} x + y + 5z = 6 \\ 2x + y + 8z = 8 \\ x + 2y + 7z = 10 \end{cases}$

```
>> M = [1,1,5,6; 2,1,8,8; 1,2,7,10];   % M = (A | b)
>> R = rref(M)
R =
    1     0     3     2
    0     1     2     4
    0     0     0     0
```

$$\Rightarrow \begin{cases} x = 2 - 3t \\ y = 4 - 2t\,, \qquad t \in \mathbb{R} \\ z = t \end{cases}$$

Rank of a matrix

- **Rank** of a matrix = the maximum number of linearly independent columns

- MATLAB command: k = rank(A)

- More advanced version: k = rank(A,tol)
  - tol: a tolerance parameter to account for numerical errors
  - The rank will be the number of singular values of A that are larger than tol

- Example:

```
>> A = [1,1,5; 2,1,8; 1,2,7];
>> k = rank(A)
k =
    2
```

- Example:

```
>> A = [1,0,0; 0,2,0; 0,0,1e-15];
>> rank(A)     % the last value is too small
ans =
    2
>> rank(A,1e-16) % specify the tolerance
ans =
    3
```

Null space of a matrix

- **Null space** of A = the set of all solutions to a system $Ax = 0$, i.e. $\{z\colon\ Az = 0\}$

- MATLAB command for finding an **orthonormal basis for the null space**:
  Z = null(A)

- Example:

```
>> M = [1,1,5,6; 2,1,8,8; 1,2,7,10];
>> Z = null(M)
Z =
    0.8725    0.1563
    0.3194   -0.8690
   -0.3564   -0.2954
    0.0984    0.3649
```

```
>> M*Z    % check whether the result is 0
ans =
   1.0e-14 *
    0.0555         0
    0.1221    0.0444
    0.0777    0.0888
```

Condition number of a matrix

- **Condition number**: $\kappa(A) = \|A^{-1}\| \, \|A\|$, where $\|\cdot\|$ is a matrix norm
  - Measure the sensitivity of the solution of $Ax = b$ to errors in the vector b
  - Useful for numerical analysis (more in MATH3230)
  - Large $\kappa \Rightarrow$ ill-conditioned

- MATLAB command: c = cond(A,p)
  - p: can be 1 (1-norm condition number)
    2 (2-norm; the default choice),
    Inf (infinity norm), or 'fro' (Frobenius norm)

- Example:

```
>> A = [1, 3; 2, 5.999];
>> c = cond(A)
c =
  4.9988e+04
```

Eigenvalues and eigenvectors

- **Eigenvalues and eigenvectors:**

$$Av = \lambda v$$

  where v is a non-zero vector

- MATLAB commands:
  - e = eig(A): returns a column vector e containing all eigenvalues of A
  - [V,D] = eig(A): returns a diagonal matrix D of eigenvalues and a matrix V whose columns are the corresponding right eigenvectors, so that $AV = VD$

- Commands for partial output (useful when handling large matrices and only some eigenvalues/eigenvectors are needed):
  - e = eigs(A,k): returns the k largest magnitude eigenvalues of A
  - [V,D] = eigs(A,k): returns the k largest magnitude eigenvalues (stored in D) and the k eigenvectors (stored in V)

Diagonalization

- If A is diagonalizable, we can use the MATLAB command [V,D] = eig(A) and get
$$A = VDV^{-1}$$

- For more general square matrices, we can consider the **Jordan canonical form**
$$A = VJV^{-1}$$
where J is a block diagonal matrix with $J = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_p \end{pmatrix}$ and each J_i is of the form
$$J_i = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & 1 & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}$$

- MATLAB command: [V,J] = jordan(A)

LU factorization

- **LU factorization:** Decompose a real square matrix A as
$$A = LU$$
where
  - L is a lower triangular matrix with all diagonal elements = 1
  - U is an upper triangular matrix
  - (Note: Mathematically, LU factorization may not exist in some cases!)

- LU is useful for solving $Ax = b$:
  1. Simplify the equation as $Ax = b \Leftrightarrow LUx = b$
  2. Solve $Ly = b$ using forward substitution
  3. Solve $Ux = y$ using backward substitution

  (Note: This is exactly one of the methods used in the backslash \ solver)

- MATLAB command: [L, U] = lu(A)

LU factorization with partial pivoting (LUP)

- **LUP factorization:** Express a real square matrix A as
$$PA = LU$$
where
  - P is a permutation matrix
  - L is a lower triangular matrix
  - U is an upper triangular matrix
  - (Note: Mathematically, LUP factorization always exists!)

- LUP is useful for solving $Ax = b$:
  1. Simplify the equation as $Ax = b \Leftrightarrow PAx = Pb \Leftrightarrow LUx = Pb$
  2. Solve $Ly = Pb$ using forward substitution
  3. Solve $Ux = y$ using backward substitution

- MATLAB command: [L, U, P] = lu(A)

LU factorization with partial pivoting (LUP)

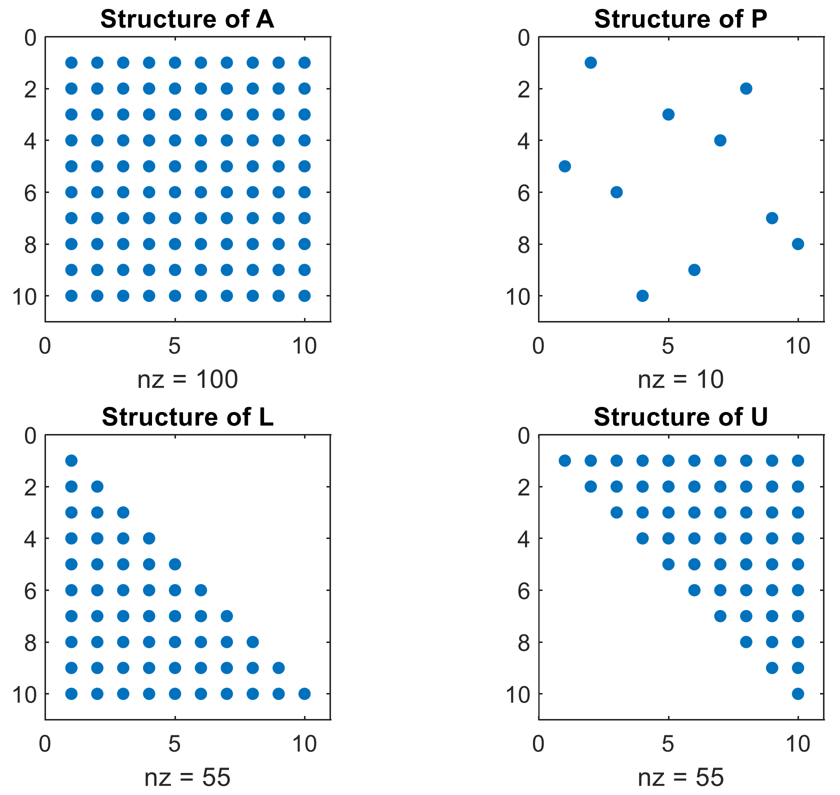
- Example:

```
A = rand(10);                % create an arbitrary 10x10 matrix
[L,U,P] = lu(A);             % perform the LUP factorization
norm(P*P'-eye(10))           % check that P is an orthogonal matrix
norm(P*A -L*U)               % check that PA = LU
```

- We can visualize the sparsity pattern of a matrix using the spy command

- Example:

```
figure;
subplot(2,2,1); spy(A); title('Structure of A');
subplot(2,2,2); spy(P); title('Structure of P');
subplot(2,2,3); spy(L); title('Structure of L');
subplot(2,2,4); spy(U); title('Structure of U');
```



QR factorization

- **QR factorization** of a real square matrix A:
$$A = QR$$
where
  - Q is an orthogonal matrix (i.e. $Q^T Q = I$)
  - R is an upper triangular matrix

- QR is also useful for solving $Ax = b$:
  1. Simplify $Ax = b \Leftrightarrow QRx = b \Leftrightarrow Q^T QRx = Q^T b \Leftrightarrow Rx = Q^T b$
  2. Solve $Rx = Q^T b$ using backward substitution

- MATLAB command: [Q,R] = qr(A)

- Example:

```
>> A = [1 2 3; 1 3 5; 7 1 8];
>> [Q,R] = qr(A)
Q =
   -0.1400   -0.5279   -0.8377
   -0.1400   -0.8270    0.5445
   -0.9802    0.1935    0.0419
R =
   -7.1414   -1.6803   -8.9618
         0   -3.3431   -4.1701
         0         0    0.5445
```

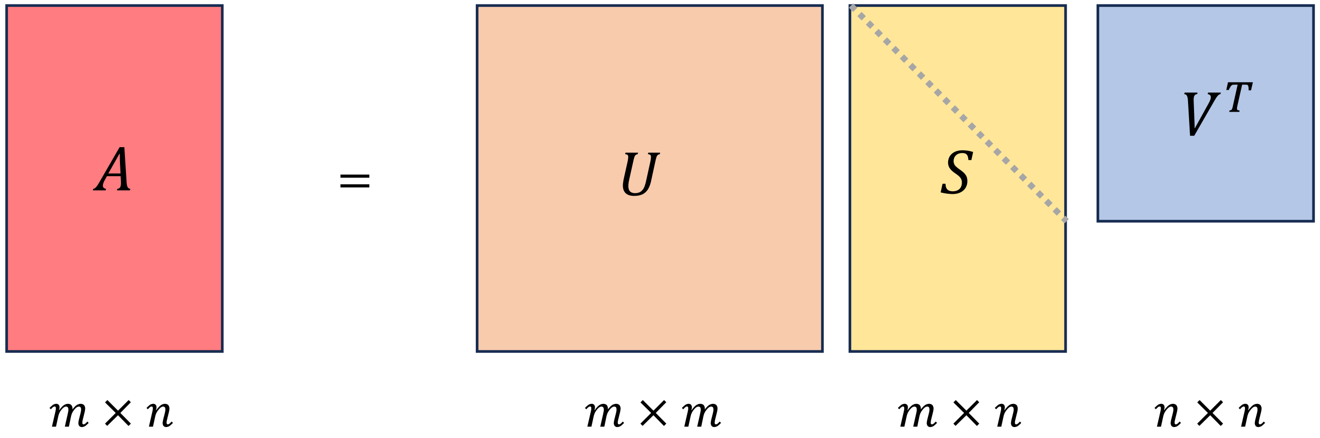
```
>> norm(Q*R-A) % verify A = QR
ans =
   3.1628e-15
>> norm(Q'*Q-eye(3))
ans =
   3.9422e-16
```

Singular value decomposition

- Analogous to diagonalization but also work for non-square matrix
- **Singular value decomposition** of any given $m \times n$ real matrix A:

$$A = USV^T$$

- Version 1 (**Full SVD**):
  - U is $m \times m$ square matrix with orthonormal columns ($U^T U = I$)
  - S is a $m \times n$ matrix with diagonal ($s_{11}, s_{22}, \dots, s_{kk}$) elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$ (the singular values) (here, $k = \min\{m, n\}$)
  - V is a $n \times n$ square matrix with orthonormal columns ($V^T V = I$)

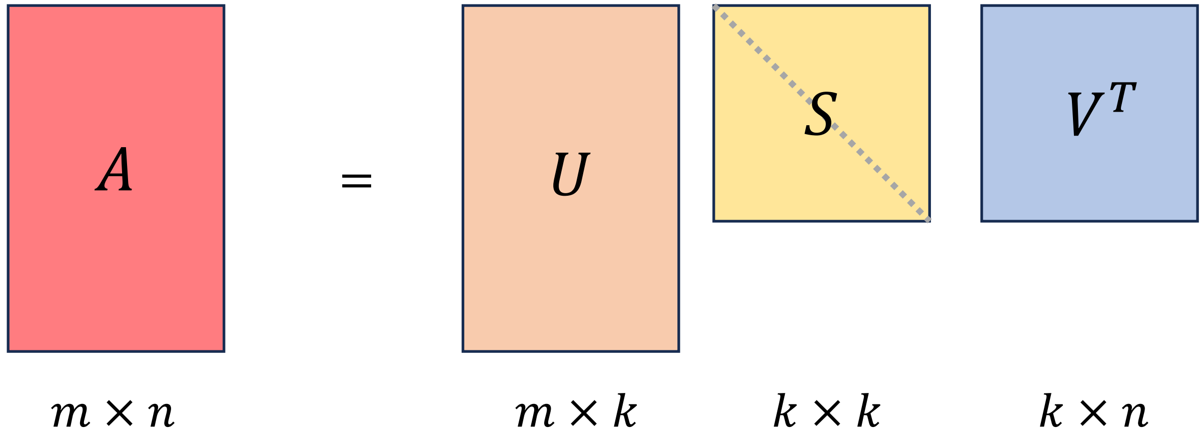


Singular value decomposition

- Analogous to diagonalization but also work for non-square matrix

- **Singular value decomposition** of any given $m \times n$ real matrix A:
$$A = USV^T$$

- Version 2 (**Reduced SVD**):
  - U is $m \times k$ matrix with orthonormal columns ($U^T U = I$) (here, $k = \min\{m, n\}$)
  - S is a $k \times k$ square matrix with diagonal elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$ (the singular values)
  - V is a $n \times k$ matrix with orthonormal columns ($V^T V = I$)



Singular value decomposition

- MATLAB commands:
  - [U,S,V] = svd(A) (Full SVD)
  - [U,S,V] = svd(A,"econ") (Reduced SVD)

- Example:

```
>> A = [1 2; 3 4; 5 6; 7 8];
>> [U,S,V] = svd(A)   % full SVD
```

U =

-0.1525	-0.8226	-0.3945	-0.3800
-0.3499	-0.4214	0.2428	0.8007
-0.5474	-0.0201	0.6979	-0.4614
-0.7448	0.3812	-0.5462	0.0407

S =

14.2691	0
0	0.6268
0	0
0	0

V =

-0.6414	0.7672
-0.7672	-0.6414

Singular value decomposition

- MATLAB commands:
  - [U,S,V] = svd(A)   (Full SVD)
  - [U,S,V] = svd(A,"econ")   (Reduced SVD)

- Example:

```
>> A = [1 2; 3 4; 5 6; 7 8];
>> [U,S,V] = svd(A,"econ")  % reduced SVD

U =
   -0.1525   -0.8226
   -0.3499   -0.4214
   -0.5474   -0.0201
   -0.7448    0.3812

S =
   14.2691         0
         0    0.6268

V =
   -0.6414    0.7672
   -0.7672   -0.6414
```

Reminder: Lab 6 this week

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	[28]	[29]	[30]	[31]	

February

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						[1]
[2]	[3]	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	1

March

Sun	Mon	Tue	Wed	Thu	Fri	Sat
2	[3]	[4]	[5]	[6]	[7]	[8]
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

April

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17		

Lecture 1- Lecture 13

Lab 1 - Lab 10 (40%)

Test 1 (30%)
Test 2 (30%)

Thank you!

Next time:

- File input/output and data analysis using MATLAB