Lecture Notes on Math for Electrical Engineers

Eleftherios Gkioulekas

These notes are constantly updated by the author. If you have not obtained this file from the author's website, it may be out of date. This notice includes the date of latest update to this file. If you are using these notes for a course, I would be very pleased to hear from you, in order to document for my University the impact of this work.

The main online lecture notes website is: http://faculty.utpa.edu/gkioulekase/

You may contact the author at: drlf@hushmail.com

Last updated: February 12, 2018

Contents

BRIEF INTRODUCTION TO LOGIC AND SETS

Basic concepts

The basic concepts we wish to introduce informally are

a) Propositions

b) Sets

c) Predicates - Quantified statements.

Propositions

- A proposition p is any statement which is true or false.
- Given two propositions p, q we define the following composite propositions.

1) Conjunction $p \wedge q$: "p is true and q is true"
   - ▸ True if both p and q are true, otherwise false.

2) Disjunction : $p \vee q$: "p is true or q is true (or both)"
   - ▸ True if at least one of the two statements p or q is true, otherwise false.

3) Negation $\overline{p}$: "p is not true"
   - ▸ True if p is false. False if p is true.

4) Exclusive Disjunction $p \veebar q$: "either p or q is true (not both)"
   - ▸ True if either p or q but not both is true. Otherwise false.

5) Implication $p \Rightarrow q$: "If p is true then q is true"
True if the truth of p implies the truth of q. Note that if p is false, then we presume that $p \Rightarrow q$ is true regardless of whether q is true or false. If p is true and q is false then $p \Rightarrow q$ is false.

6) Equivalence $p \Leftrightarrow q$: "p is true if and only if q is true"
True if p and q always have the same truth value. False if p and q have opposite truth values.

Sets

- A *set* A is an *unordered* collection of *elements*. An element can be a number, a derived object (i.e. vectors, matrices, etc.) or another set.
- A set with a finite number of elements can be defined by listing the elements.
  e.g.: $A = \{2, 3, 6, 9, 12\}$.
- *Notation*: Let A, B be sets and let x be an element.

1) $x \in A$: x belongs to A
x is an element of A

2) $x \notin A$: x does not belong to A
x is not an element of A

3) $A = B$: A and B have the same elements.

4) $A \subseteq B$: All the elements of A belong to B

- We note that: $A = B \Leftrightarrow (A \subseteq B \wedge B \subseteq A)$
- Special sets

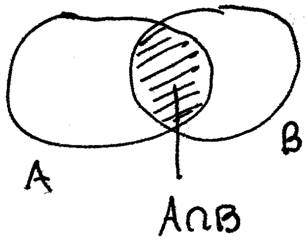
1) $\emptyset = \{\}$. The empty set.
   The empty set is the set that has no elements.
2) $\mathbb{C}$ = the set of all complex numbers
3) $\mathbb{R}$ = the set of all real numbers.
4) $\mathbb{Q}$ = the set of all rational numbers.
5) $\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$ = the set of all integers.
6) $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ = the set of all natural numbers.
7) For $n \in \mathbb{N}$: $[n] = \{1, 2, 3, \ldots, n\}$.

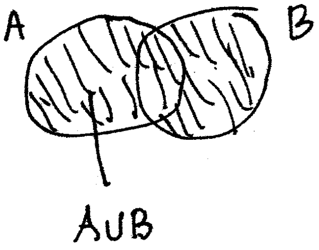
- We note that: $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R} \subseteq \mathbb{C}$
- Set operations

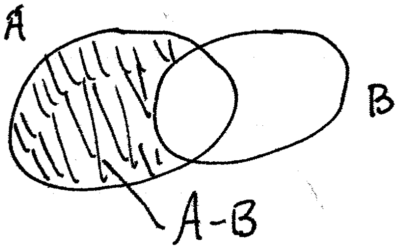
Let A, B be two sets. We define the following set operations:

1) Intersection: $A \cap B$
   $x \in A \cap B \Leftrightarrow x \in A \wedge x \in B$
2) Union: $A \cup B$
   $x \in A \cup B \Leftrightarrow x \in A \vee x \in B$
3) Difference: $A - B$
   $x \in A - B \Leftrightarrow x \in A \wedge x \notin B$

We represent these operations with Venn Diagrams as follows:







- Predicates and quantified statements
- A predicate $p(x)$ is a statement about x which is true or false depending on the value of x.
- Note that x can also be an ordered collection of elements $x = (x_1, x_2, \ldots, x_n)$. Then we write $p(x)$ as $p(x_1, x_2, \ldots, x_n)$.
- Given a predicate $p(x)$ and a set A, we define the following quantified statements:

1) $\forall x \in A : p(x)$

   For all $x \in A$, $p(x)$ is satisfied.

2) $\exists x \in A : p(x)$

   There is at least one $x \in A$ such that $p(x)$ is satisfied.

3) $\exists! \, x \in A : p(x)$

   There is a unique $x \in A$ such that $p(x)$ is satisfied.

- If A is a finite set, then the above quantified statements are abbreviations for conjunction, disjunction, and exclusive disjunction: For example:

$$(\forall x \in \{a, b, c\} : p(x)) \Leftrightarrow (p(a) \wedge p(b) \wedge p(c))$$

$$(\exists x \in \{a, b, c\} : p(x)) \Leftrightarrow (p(a) \vee p(b) \vee p(c))$$

$$(\exists! \, x \in \{a, b, c\} : p(x)) \Leftrightarrow (p(a) \veebar p(b) \veebar p(c))$$

- Quantifiers can be nested to give compound quantified statements. For example:

1) $\forall x \in A : \exists y \in B : p(x, y)$

   For all $x \in A$, there is a $y \in B$, such that $p(x, y)$ is satisfied.

2) $\exists x \in A : \forall y \in B : p(x,y)$

There is an $x \in A$ such that for all $y \in B$, $p(x,y)$ is satisfied.

- Important quantified statements from algebra

$\forall a,b \in \mathbb{R} : (ab = 0 \Leftrightarrow a = 0 \lor b = 0)$

$\forall a,b \in \mathbb{R} : (a^2 + b^2 = 0 \Leftrightarrow a = 0 \land b = 0)$

$\forall a,b \in \mathbb{R} : (|a| + |b| = 0 \Leftrightarrow a = 0 \land b = 0)$

- <u>Definitions of sets</u>

There are 3 methods for defining sets:

1) <u>By listing</u>: For finite sets we can simply list the elements.

e.g.: $A = \{3, 7, 10, 12\}$

2) <u>By predicate</u>: $A = \{x \in U \mid p(x)\}$

with U a predefined set and $p(x)$ a predicate.

Belonging condition: $x \in A \Leftrightarrow (x \in U \land p(x))$

e.g.: We can use definition by predicate to define intervals:

$[a,b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$

$(a,b) = \{x \in \mathbb{R} \mid a < x < b\}$

$[n] = \{x \in \mathbb{N} \mid 1 \leq x \leq n\} = \{1, 2, \dots, n\}$

3) <u>By mapping</u>: $A = \{\varphi(x) \mid x \in U \land p(x)\}$

with $\varphi(x)$ some <u>expression</u> of x, U a predefined set, and $p(x)$ a <u>predicate</u>.

Belonging condition: $y \in A \Leftrightarrow \exists x \in U : (\varphi(x) = y \land p(x))$

EXAMPLES

a) The set of complex numbers:

$\mathbb{C} = \{a+bi \mid a,b \in \mathbb{R}\}$.

$z \in \mathbb{C} \Leftrightarrow \exists a,b \in \mathbb{R} : z = a+bi$

b) The set of rational numbers:

$\mathbb{Q} = \{a/b \mid a \in \mathbb{Z} \wedge b \in \mathbb{N}-\{0\}\}$

$x \in \mathbb{Q} \Leftrightarrow \exists a \in \mathbb{Z} : \exists b \in \mathbb{N}-\{0\} : x = a/b$.

c) The set of even integers

$A = \{2k \mid k \in \mathbb{Z}\}$

$x \in A \Leftrightarrow \exists k \in \mathbb{Z} : x = 2k$

d) The set of odd integers

$A = \{2k+1 \mid k \in \mathbb{Z}\}$

$x \in A \Leftrightarrow \exists k \in \mathbb{Z} : x = 2k+1$.

e) $A = \{a^2+b^2 \mid a,b \in \mathbb{R} \wedge a+3b < 1\}$

$x \in A \Leftrightarrow \exists a,b \in \mathbb{R} : (x = a^2+b^2 \wedge a+3b < 1)$

- Cartesion product

We use definition by mapping to define the cartesian product between sets.

- An ordered pair (a,b) is an ordered collection of two elements a and b. We call a and b the components of (a,b).
- We note that : $(a,b) = (c,d) \Leftrightarrow (a = c \wedge b = d)$.

- Let A,B be two sets. We define the Cartesian product
$A\times B = \{(a,b) \mid a\in A \wedge b\in B\}$.
We also define:
$A^2 = A\times A = \{(a,b) \mid a\in A \wedge b\in A\}$

EXAMPLE

For $A=\{1,2,3\}$ and $B=\{5,6\}$. Calculate $A\times B$, A^2, B^2.

Solution

$A\times B = \{1,2,3\}\times\{5,6\} =$
$= \{(1,5),(1,6),(2,5),(2,6),(3,5),(3,6)\}$

$A^2 = A\times A = \{1,2,3\}\times\{1,2,3\} =$
$= \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)\}$

$B^2 = B\times B = \{5,6\}\times\{5,6\} =$
$= \{(5,5),(5,6),(6,5),(6,6)\}$

↳ The above can be generalized as follows

- An ordered n-tuplet $(x_1,x_2,\ldots,x_n)$ is an ordered collection of n elements $x_1,x_2,\ldots,x_n$.
- Let $x=(x_1,x_2,\ldots,x_n)$ and $y=(y_1,y_2,\ldots,y_n)$.
We note that:
$x=y \iff \forall a\in[n] : x_a = y_a$
- Let $A_1,A_2,\ldots,A_n$ be n sets. We define:
$A_1\times A_2\times\cdots\times A_n = \{(x_1,x_2,\ldots,x_n) \mid \forall a\in[n] : x_a\in A_a\}$
- Special case:
$A_1\times A_2\times A_3 = \{(x_1,x_2,x_3) \mid x_1\in A_1 \wedge x_2\in A_2 \wedge x_3\in A_3\}$.

EXERCISES

① Let $A=[7]$, $B=\{x\in A \mid x>4\}$, and $C=\{x-1 \mid x\in B\}$.
List the elements of
a) B b) C c) $B\cap C$ d) $B\cup C$
e) $A-B$ f) $B-C$ g) $C-B$

② Write out the following statements in English

a) $\forall a\in A: \exists b\in B: (a,b)\in f$
b) $\exists a\in A: \forall b\in B: a+b>3$
c) $\forall a\in A: \exists b\in B: (ab>2 \wedge a+b>1)$
d) $\forall a,b\in A: \exists c\in B: \forall d\in A: ab+bd<3$
e) $\exists a\in A: \forall b\in B: (ab>3 \Rightarrow b>2)$
f) $\forall a\in A: \exists b\in B: (3a>b \veebar a+b<0)$

③ Write the following statements symbolically using quantifiers.

a) Every real number is equal to itself.
b) There is a real number x such that $3x-1=2(x+3)$
c) For every real number x, there is a natural number n such that $n>x$.
d) For every real number x, there is a complex number y such that $y^2=x$.
e) There is a real number x such that for all real numbers y we have $x+y=0$.

f) For all $\varepsilon > 0$, there is a $\delta > 0$ such that for all real numbers x, if $x_0 - \delta < x < x_0 + \delta$ then $|f(x) - a| < \varepsilon$.

g) There is a real number b such that for all natural numbers n we have $a_n < b$.

h) For all $\varepsilon > 0$, there is a natural number n_0 such that for any two natural numbers n_1 and n_2, if $n_1 > n_0$ and $n_2 > n_0$, then $|a_{n_1} - a_{n_2}| < \varepsilon$.

i) For any $M > 0$, there is a natural number n_0, such that for any other natural number n, if $n > n_0$ then $\alpha_n > M$.

④ Write the belonging condition $x \in A$ for the following sets, using quantifiers.

a) $A = \{x^2 + 1 \mid x \in \mathbb{Q} \wedge 2x < 1\}$

b) $A = \{3x + 1 \mid x \in \mathbb{Z} \wedge x \text{ is a prime number}\}$

c) $A = \{x \in \mathbb{R} \mid x^2 + 3x > 0\}$

d) $A = \{a^3 + b^3 + c^3 \mid a, b \in \mathbb{R} \wedge c \in \mathbb{Q} \wedge a + b + c = 0\}$

e) $A = \{x \in \mathbb{R} \mid x^2 + 2x < 0 \vee 3x + 1 > -4 + x\}$

f) $A = \{a^2 - b^2 \mid a \in \mathbb{N} \wedge b \in \mathbb{R} \wedge a + b > 5\}$

g) $A = \{x \in \mathbb{Z} \mid \exists k \in \mathbb{Z} : x = 3k\}$

h) $A = \{ab \mid a, b \in \mathbb{R} \wedge (a + b > 2 \vee a - b < -3)\}$

i) $A = \{x \in \mathbb{R} \mid \exists y \in \mathbb{R} : y^2 + y = x\}$

j) $A = \{x \in \mathbb{R} \mid \forall y \in \mathbb{R} : x < y^2 + 1\}$

k) $A = \{a + b \mid a, b \in \mathbb{R} \wedge (ab > 1 \Rightarrow a^2 + b^2 > 2)\}$

l) $A = \{abc \mid a, b, c \in \mathbb{R} \wedge (a + b > 2 \underline{\vee}\, a - c < 3)\}$

m) $A = \{2a + 3b \mid a, b \in \mathbb{R} \wedge ab > 1 \wedge a - b < 0\}$

⑤ List the elements for the following cartesian products

a) $A \times B$ with $A = \{2,3,4\}$ and $B = \{7,8\}$

b) $A \times B$ with $A = \{1\}$ and $B = \{3,9\}$

c) $A \times B$ with $A = \{3\}$ and $B = \{5\}$

d) $[2] \times [3]$

e) $A \times B$ with $A = [5] - [2]$ and $B = [2] \cap [4]$

f) $A \times B \times C$ with $A = [3] - \{1\}$, $B = [3] \cap [6]$, and $C = [2]$.

g) $A \times B \times C$ with $A = \{2\}$, $B = [2]$, $C = [4] - [2]$.

LINEAR ALGEBRA

▼ Matrices – Definitions

- An $n\times m$ matrix A is a collection of nm numbers $A_{ab}\in\mathbb{R}$ (with $a\in[n]$ and $b\in[m]$) arranged in n rows and m columns as follows:

$$A=\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix}\Bigg\downarrow \text{ rows } a=1,2,\ldots,n$$

$$\xrightarrow{\qquad\qquad\qquad}$$

columns $b=1,2,\ldots,m$

Remember:
A_{rc} : row, column
A_{vh} : vertical, horizontal

We also write $A=[A_{ab}]$.

A_{ab} = the element of A at row a and column b.

- $M_{nm}(\mathbb{R})$ = the set of all $n\times m$ matrices with elements from $\mathbb{R}$.
- For $n=m$, an $n\times n$ matrix is called a square matrix and we write $M_n(\mathbb{R})=M_{nn}(\mathbb{R})$.
- Let $A,B\in M_{nm}(\mathbb{R})$ be two matrices, Then $A=B \Leftrightarrow \forall a\in[n]:\forall b\in[m]: A_{ab}=B_{ab}$.

▶ Zero matrix :

Let $A\in M_{nm}(\mathbb{R})$ be a matrix. Then $A=\mathbb{0} \Leftrightarrow \forall a\in[n]:\forall b\in[m]: A_{ab}=0$

► Identity Matrix

We say that $I \in M_n(\mathbb{R})$ is an identity matrix if and only if
$$\forall a, b \in [n]: \ I_{ab} = \begin{cases} 1 \ , & \text{if } a=b \\ 0 \ , & \text{if } a \neq b \end{cases}$$

▼ Basic operations with matrices

- Let $A, B, C \in M_{nm}(\mathbb{R})$ be given matrices, and let $\lambda \in \mathbb{R}$.
  Then, we define:
  $C = A + B \Leftrightarrow \forall a \in [n] : \forall b \in [m] : \ C_{ab} = A_{ab} + B_{ab}$ (addition)
  $C = \lambda A \Leftrightarrow \forall a \in [n] : \forall b \in [m] : \ C_{ab} = \lambda A_{ab}$. (scalar multiplication)
  We also define: $-A = (-1)A$ and $A - B = A + (-1)B$.
- Properties of matrix addition:
  $\forall A, B \in M_{nm}(\mathbb{R}) : \ A + B = B + A$
  $\forall A, B, C \in M_{nm}(\mathbb{R}) : \ (A+B) + C = A + (B+C)$
  $\forall A \in M_{nm}(\mathbb{R}) : \ A + \mathbf{0} = \mathbf{0} + A = A$
  $\forall A \in M_{nm}(\mathbb{R}) : \exists B \in M_{nm}(\mathbb{R}) : A + B = B + A = \mathbf{0}$
- Properties of scalar multiplication
  $\forall \lambda \in \mathbb{R} : \forall A, B \in M_{nm}(\mathbb{R}) \ : \lambda(A+B) = \lambda A + \lambda B$
  $\forall \lambda, \mu \in \mathbb{R} : \forall A \in M_{nm}(\mathbb{R}) : \ \lambda(\mu A) = (\lambda\mu) A$
  $\forall \lambda, \mu \in \mathbb{R} : \forall A \in M_{nm}(\mathbb{R}) : \ (\lambda + \mu) A = \lambda A + \mu A$
  $\forall A \in M_{nm}(\mathbb{R}) : \ 1 \cdot A = A$
  $\forall \lambda \in \mathbb{R} : \ \lambda \mathbf{0} = \mathbf{0}$
  $\forall A \in M_{nm}(\mathbb{R}) : \ (-1) A = -A$

EXAMPLES

a) Let $A = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 1 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 3 & 1 \\ -1 & 0 & -2 \end{bmatrix}$.

Calculate $A+B$ and $2A-3B$.

Solution

$$A+B = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 1 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 3 & 1 \\ -1 & 0 & -2 \end{bmatrix} =$$

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

$$2A-3B = 2\begin{bmatrix} 1 & 3 & 2 \\ 3 & 1 & 4 \end{bmatrix} - 3\begin{bmatrix} 2 & 3 & 1 \\ -1 & 0 & -2 \end{bmatrix} =$$

$$= \begin{bmatrix} 2 & 6 & 4 \\ 6 & 2 & 8 \end{bmatrix} - \begin{bmatrix} 6 & 9 & 3 \\ -3 & 0 & -6 \end{bmatrix} =$$

$$= \begin{bmatrix} 2-6 & 6-9 & 4-3 \\ 6-(-3) & 2-0 & 8-(-6) \end{bmatrix} = \begin{bmatrix} -4 & -3 & 1 \\ 9 & 2 & 14 \end{bmatrix}$$

b) Prove: $\forall A, B, C \in M_{nm}(\mathbb{R}): (A+B)+C = A+(B+C)$

Solution

Let $A, B, C \in M_{nm}(\mathbb{R})$ be given. Let $a \in [n]$ and $b \in [m]$ be given. Then:

$$[(A+B)+C]_{ab} = (A+B)_{ab} + C_{ab} = (A_{ab} + B_{ab}) + C_{ab} =$$

$$= A_{ab} + (B_{ab} + C_{ab}) = A_{ab} + (B+C)_{ab} =$$

$$= [A+(B+C)]_{ab}.$$

It follows that

$\forall a \in [n] : \forall b \in [m] : [(A+B)+C]_{ab} = [A+(B+C)]_{ab}$

$\Rightarrow (A+B)+C = A+(B+C)$

and therefore:

$\forall A, B, C \in M_{nm}(\mathbb{R}) : (A+B)+C = A+(B+C)$.

c) Prove: $\forall \lambda, \mu \in \mathbb{R} : \forall A \in M_{nm}(\mathbb{R}) : \lambda(\mu A) = (\lambda\mu)A$

Solution

Let $\lambda, \mu \in \mathbb{R}$ and $A \in M_{nm}(\mathbb{R})$ be given. Let $a \in [n]$ and $b \in [m]$ be given. Then

$[\lambda(\mu A)]_{ab} = \lambda(\mu A)_{ab} = \lambda(\mu A_{ab}) = (\lambda\mu)A_{ab} = [(\lambda\mu)A]_{ab}$.

It follows that

$\forall a \in [n] : \forall b \in [m] : [\lambda(\mu A)]_{ab} = [(\lambda\mu)A]_{ab}$

$\Rightarrow \lambda(\mu A) = (\lambda\mu)A$

and therefore

$\forall \lambda, \mu \in \mathbb{R} : \forall A \in M_{nm}(\mathbb{R}) : \lambda(\mu A) = (\lambda\mu)A$.

EXERCISES

① Let A, B be the matrices

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -3 & 1 & -1 \\ 2 & -4 & 2 \end{bmatrix}$$

a) Evaluate $G = 3A - 2B$

b) Solve with respect to X the equation $2A + 3(X - B) = A + B$

② Let A, B be the matrices

$$A = \begin{bmatrix} a & -2a & c \\ 0 & -a & b \\ a+b & 0 & -1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 1 & 2a & c \\ a & b-a & -b \\ a-b & 0 & -1 \end{bmatrix}$$

Evaluate and simplify $G = A + B$ and $D = 2(A - B) - (A + 2B)$

③ Consider the matrix-valued functions

$$A(x) = \begin{bmatrix} 1 & x^2 \\ x & 3x \end{bmatrix}, \quad \forall x \in \mathbb{R}$$

$$B(x) = \begin{bmatrix} x-1 & 2x \\ x^2 & 1 \end{bmatrix}, \quad \forall x \in \mathbb{R}$$

a) Evaluate and simplify the function $G(x) = 2A(2x+1) - B(x-2), \forall x \in \mathbb{R}$

b) Solve with respect to $Y(x)$ the matrix equation $3A(x) + 2(Y(x) + A(x)) = A(x+1) - B(x)$

④ Given the function

$$A(x) = \begin{bmatrix} 1 & 2x & x^2 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{bmatrix}$$

Show that

$$A(3x) + 3A(x) = A(0) + 3A(2x), \quad \forall x \in \mathbb{R}$$

⑤ Prove that

a) $\forall \lambda \in \mathbb{R} : \forall A, B \in M_{nm}(\mathbb{R}) : \lambda(A+B) = \lambda A + \lambda B$

b) $\forall \lambda, \mu \in \mathbb{R} : \forall A \in M_{nm}(\mathbb{R}) : (\lambda + \mu) A = \lambda A + \mu A$

▼ Matrix multiplication

The product AB of two matrices A, B can be defined only when $A \in M_{nl}(\mathbb{R})$ and $B \in M_{lm}(\mathbb{R})$. That is, the number of columns of A must be equal to the number of rows of B. Then we define the product as follows:

- For $A \in M_{nl}(\mathbb{R})$ and $B \in M_{lm}(\mathbb{R})$, we define $(AB) \in M_{nm}(\mathbb{R})$ such that

$$\boxed{\forall a \in [n] : \forall b \in [m] : \ (AB)_{ab} = \sum_{\gamma=1}^{l} A_{a\gamma} B_{\gamma b}}$$

↦ To illustrate the definition, we consider the following special cases:

a) Row matrix X Column matrix: $A \in M_{1n}(\mathbb{R}) \land B \in M_{n1}(\mathbb{R})$.

Then $AB \in M_1(\mathbb{R})$ with

$$AB = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} =$$

$$= \begin{bmatrix} a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \end{bmatrix}$$

b) Product of 2×2 matrices: $A, B \in M_2(\mathbb{R})$.

$$AB = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} c_1 & c_2 \\ d_1 & d_2 \end{bmatrix} = \begin{bmatrix} a_1 c_1 + a_2 d_1 & a_1 c_2 + a_2 d_2 \\ b_1 c_1 + b_2 d_1 & b_1 c_2 + b_2 d_2 \end{bmatrix}$$

↳ From the above examples we see that the element $(AB)_{ab}$ is the product of row a of matrix A and column b of matrix B.

▶ Properties of Matrix Multiplication

$\forall A \in M_{nk}(\mathbb{R}) : \forall B \in M_{kl}(\mathbb{R}) : \forall C \in M_{lm}(\mathbb{R}) : (AB)C = A(BC)$

$\forall A \in M_{nk}(\mathbb{R}) : \forall B, C \in M_{km}(\mathbb{R}) : \quad A(B+C) = AB + AC$

$\forall B, C \in M_{nk}(\mathbb{R}) : \forall A \in M_{km}(\mathbb{R}) : \quad (B+C)A = BA + CA$

$\forall \lambda \in \mathbb{R} : \forall A \in M_{nk}(\mathbb{R}) : \forall B \in M_{km}(\mathbb{R}) : \lambda(AB) = (\lambda A)B = A(\lambda B)$

$\forall A \in M_n(\mathbb{R}) : IA = AI = A \quad$ ($I \in M_n(\mathbb{R})$ is the identity matrix)

$\forall A \in M_n(\mathbb{R}) : A\mathbf{0} = \mathbf{0}A = \mathbf{0}$

- It is not true for all matrices that $AB = BA$ (see homework for a counterexample). This creates some interesting complications.

▶ Manipulation Properties

$\forall A, B, C \in M_{nm}(\mathbb{R}) : A = B \Leftrightarrow A + C = B + C$

$\forall A, B \in M_{nk}(\mathbb{R}) : \forall C \in M_{km}(\mathbb{R}) : A = B \Rightarrow AC = BC$

$\forall C \in M_{nk}(\mathbb{R}) : \forall A, B \in M_{km}(\mathbb{R}) : A = B \Rightarrow CA = CB$

$\forall A, B, C \in M_{nm}(\mathbb{R}) : A + B = C \Leftrightarrow A = C - B$

- Note that the cancellation property $CA = CB \Rightarrow A = B$ is not true for all matrices

▶ Matrix powers

Let $A \in M_n(\mathbb{R})$ be a square matrix. We define

$$A^n = \underbrace{A \cdot A \cdot \ldots \cdot A}_{n \text{ times}}$$

$\forall a, b \in \mathbb{N} - \{0\} : \forall A \in M_n(\mathbb{R}) : A^a A^b = A^{a+b}$

$\forall a, b \in \mathbb{N} - \{0\} : \forall A \in M_n(\mathbb{R}) : (A^a)^b = A^{ab}$

EXAMPLES

a) Let $A = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$. Find all $x, y \in \mathbb{R}$ such that $A^2 = xA - yI$.

Solution

We note that

$$A^2 = AA = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 2\cdot 2 + 1\cdot 3 & 2\cdot 1 + 1\cdot 2 \\ 3\cdot 2 + 2\cdot 3 & 3\cdot 1 + 2\cdot 2 \end{bmatrix} =$$

$$= \begin{bmatrix} 4+3 & 2+2 \\ 6+6 & 3+4 \end{bmatrix} = \begin{bmatrix} 7 & 4 \\ 12 & 7 \end{bmatrix}$$

and

$$xA - yI = x\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix} - y\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2x & x \\ 3x & 2x \end{bmatrix} - \begin{bmatrix} y & 0 \\ 0 & y \end{bmatrix} =$$

$$= \begin{bmatrix} 2x-y & x \\ 3x & 2x-y \end{bmatrix}$$

It follows that

$$A^2 = xA - yI \Leftrightarrow \begin{bmatrix} 7 & 4 \\ 12 & 7 \end{bmatrix} = \begin{bmatrix} 2x-y & x \\ 3x & 2x-y \end{bmatrix} \Leftrightarrow \begin{cases} 2x-y=7 \\ 3x=12 \\ x=4 \end{cases}$$

$$\Leftrightarrow \begin{cases} 2\cdot 4 - y = 7 \\ x=4 \end{cases} \Leftrightarrow \begin{cases} 8-y=7 \\ x=4 \end{cases} \Leftrightarrow \begin{cases} y = 8-7 = 1 \\ x=4 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow x=4 \wedge y=1.$$

b) Let $A, B \in M_n(\mathbb{R})$ such that $A^2 = I$ and $B^2 = B$. Show that $(2B-I)^2 = I$ and $(A+I)^2 = 2(A+I)$.

Solution

Assume that $A, B \in M_n(\mathbb{R})$ with $A^2 = I$ and $B^2 = B$. Then

$$\begin{aligned}
(2B-I)^2 &= (2B-I)(2B-I) = 2B(2B-I) - I(2B-I) = \\
&= (2B)(2B) - (2B)I - I(2B) + I^2 = \\
&= 4B^2 - 2B - 2B + I = 4B^2 - 4B + I \overset{*}{=} \\
&= 4B - 4B + I = 0B + I = 0 + I = I.
\end{aligned}$$

and

$$\begin{aligned}
(A+I)^2 &= (A+I)(A+I) = A(A+I) + I(A+I) = \\
&= AA + AI + IA + I^2 = A^2 + A + A + I = \\
&= A^2 + 2A + I. \overset{*}{=} I + 2A + I = 2A + 2I \\
&= 2(A+I).
\end{aligned}$$

c) Prove: $\forall B, C \in M_{nk}(\mathbb{R}) : \forall A \in M_{km}(\mathbb{R}) : (B+C)A = BA + CA$

Solution

Let $B, C \in M_{nk}(\mathbb{R})$ and $A \in M_{km}(\mathbb{R})$ be given. Let $a \in [n]$ and $b \in [m]$ be given. Then

$$\begin{aligned}
[(B+C)A]_{ab} &= \sum_{\gamma \in [k]} (B+C)_{a\gamma} A_{\gamma b} = \sum_{\gamma \in [k]} (B_{a\gamma} + C_{a\gamma}) A_{\gamma b} = \\
&= \sum_{\gamma \in [k]} (B_{a\gamma} A_{\gamma b} + C_{a\gamma} A_{\gamma b}) = \\
&= \sum_{\gamma \in [k]} B_{a\gamma} A_{\gamma b} + \sum_{\gamma \in [k]} C_{a\gamma} A_{\gamma b} = \\
&= (BA)_{ab} + (CA)_{ab} = (BA + CA)_{ab}
\end{aligned}$$

It follows that

$\forall a \in [n] : \forall b \in [m] : [(B+C)A]_{ab} = (BA+CA)_{ab} \Rightarrow$

$\Rightarrow (B+C)A = BA+CA$

and therefore

$\forall : B, C \in M_{nk}(\mathbb{R}) : \forall A \in M_{km}(\mathbb{R}) : (B+C)A = BA + CA.$

d) Prove: $\forall A, B, C \in M_{nm}(\mathbb{R}) : (A = B \Rightarrow A+C = B+C)$

Solution

Let $A, B, C \in M_{nm}(\mathbb{R})$ be given and assume that $A=B$.

Then:

$A = B \Rightarrow \forall a \in [n] : \forall b \in [m] : A_{ab} = B_{ab} \qquad (1)$

Let $a \in [n]$ and $b \in [m]$ be given. Then:

$(A+C)_{ab} = A_{ab} + C_{ab} = B_{ab} + C_{ab} = (B+C)_{ab}$

and it follows that

$\forall a \in [n] : \forall b \in [m] : (A+C)_{ab} = (B+C)_{ab}$

$\Rightarrow A+C = B+C$

and therefore we have shown that

$\forall A, B, C \in M_{nm}(\mathbb{R}) : (A = B \Rightarrow A+C = B+C)$

EXERCISES

(6) Consider the matrix

$$A = \begin{bmatrix} 2 & 5 \\ 3 & 1 \end{bmatrix}$$

a) Find the unique $x, y \in \mathbb{R}$ such that $A^2 = xA + yI$

b) Use (a) to find $z, w \in \mathbb{R}$ such that $A^3 = zA + wI$.

(7) Given the matrices

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 2 & 1 & -1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 3 & -1 & 0 \\ 1 & 2 & -1 \\ 3 & 1 & 0 \end{bmatrix}$$

evaluate and simplify

a) $C = AB$ c) $E = 2A^3 - 3A + I$

b) $D = BA - 3B^2$

(8) Given the matrices

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} a & b & c \\ c & a & b \\ b & c & a \end{bmatrix}$$

evaluate and simplify $C = AB - BA$

(9) Given the matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ with $a, b, c, d \in \mathbb{R}$

show that: $A^2 - (a+d)A + (ad - bc)I = 0$

(10) Prove the following properties

a) $\forall A \in M_{nk}(\mathbb{R}) : \forall B, C \in M_{km}(\mathbb{R}) : \; A(B+C) = AB + AC$

b) $\forall A \in M_{nk}(\mathbb{R}) : \forall B \in M_{k\ell}(\mathbb{R}) : \forall C \in M_{\ell m}(\mathbb{R}) : \; (AB)C = A(BC)$

(11) Rotation matrix

Let $R(\vartheta) = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}$

Show that

a) $R(\vartheta_1)R(\vartheta_2) = R(\vartheta_1 + \vartheta_2)$, $\forall \vartheta_1, \vartheta_2 \in \mathbb{R}$

b) $R(\vartheta)R(-\vartheta) = I$, $\forall \vartheta \in \mathbb{R}$.

(12) For $A = \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}$ and $B = \begin{bmatrix} 3 & 4 \\ 2 & 7 \end{bmatrix}$

show that $AB \neq BA$

(13) For $A = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$ with $a, b \in \mathbb{R}$, show that $AB = BA$.

(14) Consider the function

$$\forall x \in \mathbb{R} : M(x) = \begin{bmatrix} 1 & 0 & x \\ -x & 1 & -x^2/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Show that

$\forall a, b \in \mathbb{R} : \; M(a)M(b) = M(a+b)$.

(15) Let $z = a + bi \in \mathbb{C}$ be a complex number, and define

$$M(z) = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

Show that

a) $\forall z_1, z_2 \in \mathbb{C} : M(z_1 + z_2) = M(z_1) + M(z_2)$

b) $\forall z_1, z_2 \in \mathbb{C} : M(z_1 z_2) = M(z_1) M(z_2)$

$\hookrightarrow$ This shows that $M(z)$ "imitates" the behaviour of complex number algebra.

(16) Let $A, B \in M_n(\mathbb{R})$. Show that

a) $AB = BA \Rightarrow (A - \lambda I)(B - \lambda I) = (B - \lambda I)(A - \lambda I), \forall \lambda \in \mathbb{R}$.

b) $(A+B)^2 = A^2 + 2AB + B^2 \Rightarrow AB = BA$

c) $A^2 = A \Rightarrow (A - I)^2 = I - A$

d) $AB = BA \Rightarrow A^2 B^2 = B^2 A^2$

e) $(B^2 = I \wedge AB = -AB) \Rightarrow AB = BA = \mathbf{0}$

(17) Let $A \in M_n(\mathbb{R})$ such that $A^2 = I$. Show that the matrices

$B = (1/2)(I + A)$

$C = (1/2)(I - A)$

satisfy $B^2 = B$ and $C^2 = C$.

Matrix Inverses

- Let $A \in M_n(\mathbb{R})$ be a square matrix. We say that
  B inverse of $A \iff AB = BA = I$.
- We define the set of all matrices $A \in M_n(\mathbb{R})$ that have an inveuse as:
  $$GL(n, \mathbb{R}) = \{ A \in M_n(\mathbb{R}) \mid \exists B \in M_n(\mathbb{R}) : AB = BA = I \}$$
  We say that
  A non-singular $\iff A \in GL(n, \mathbb{R})$
  A singular $\iff A \notin GL(n, \mathbb{R})$
- Square matrices are not guaranteed to have an inveuse. For example, for $A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$, we have:
  $$\forall x, y, z, w \in \mathbb{R} : \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ z & w \end{bmatrix} \neq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
  However, we will argue that if a matrix does have an inverse, it is unique:
- $$\boxed{\forall A, B, C \in M_n(\mathbb{R}) : \begin{cases} AB = BA = I \\ AC = CA = I \end{cases} \implies B = C}$$

Proof

Let $A, B, C \in M_n(\mathbb{R})$ be given such that $AB = BA = I$ and $AC = CA = I$. Then

$$\begin{aligned} B &= BI && [\text{Identity matrix}] \\ &= B(AC) && [\text{Hypothesis } AC = I] \\ &= (BA)C && [\text{Associative property}] \end{aligned}$$

$= IC$ [Hypothesis $BA = I$]

$= C$ [Identity matrix]

It follows that

$$\forall A, B, C \in M_n(\mathbb{R}) : \begin{cases} AB = BA = I \\ AC = CA = I \end{cases} \Rightarrow B = C \qquad \square$$

↳ The unique inverse of A is denoted as A^{-1}, as long as it exists.

▶ Cancellation property.

$$\boxed{\begin{array}{l} \forall A, B \in M_n(\mathbb{R}) : \forall C \in GL(n, \mathbb{R}) : (CA = CB \Leftrightarrow A = B) \\ \forall A, B \in M_n(\mathbb{R}) : \forall C \in GL(n, \mathbb{R}) : (AC = BC \Leftrightarrow A = B) \end{array}}$$

Proof

We show only the first statement. Let $A, B \in M_n(\mathbb{R})$ and $C \in GL(n, \mathbb{R})$ be given such that $CA = CB$. Then

$A = IA =$ [identity matrix]

$= (C^{-1}C)A$ [C^{-1} inverse of C]

$= C^{-1}(CA)$ [associative property]

$= C^{-1}(CB)$ [hypothesis: $CA = CB$]

$= (C^{-1}C)B$ [associative property]

$= IB$ [C^{-1} inverse of C]

$= B$ [identity matrix]

It follows that
$\forall A,B \in M_n(\mathbb{R}) : \forall C \in GL(n,\mathbb{R}) : (CA = CB \Longrightarrow A = B)$

► Inverse of a 2x2 matrix

- Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in M_2(\mathbb{R})$ be a 2x2 square matrix

a) If $D = ad - bc \neq 0$, then A is non-singular with

$$A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

b) If $D = ad - bc = 0$, then A is singular.

► Application to 2x2 linear systems.

Any 2x2 linear system given by

$$\begin{cases} a_{11}x + a_{12}y = b_1 \\ a_{21}x + a_{22}y = b_2 \end{cases}$$

can be rewritten in terms of matrix algebra as:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

and then solved using the following property:

- $\boxed{\forall A \in GL(n,\mathbb{R}) : \forall x, b \in M_{n1}(\mathbb{R}) : (Ax = b \Leftrightarrow x = A^{-1}b)}$

Proof

Let $A \in GL(n,\mathbb{R})$ and $x,b \in M_{n1}(\mathbb{R})$ be given. Then:

$$\begin{aligned} Ax=b &\Leftrightarrow A^{-1}(Ax) = A^{-1}b && \text{[cancellation property]} \\ &\Leftrightarrow (A^{-1}A)x = A^{-1}b && \text{[associative property]} \\ &\Leftrightarrow Ix = A^{-1}b && \text{[A^{-1} inverse of A]} \\ &\Leftrightarrow x = A^{-1}b && \text{[identity matrix]} \end{aligned}$$

It follows that

$$\forall A \in GL(n,\mathbb{R}) : \forall x,b \in M_{n1}(\mathbb{R}) : (Ax=b \Leftrightarrow x = A^{-1}b) \qquad \square$$

Consequently, if $a_{11}a_{22}-a_{12}a_{21} \neq 0$, then:

$$\begin{cases} a_{11}x+a_{12}y=b_1 \\ a_{21}x+a_{22}y=b_2 \end{cases} \Leftrightarrow \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \Leftrightarrow$$

$$\Leftrightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1}\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} =$$

$$= \frac{1}{a_{11}a_{22}-a_{12}a_{21}}\begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

from which we may calculate the unique solutions for (x,y).

↳ Notation : 2x2 determinant

The expression $ad-bc$ is the 2x2 determinant of the matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and we write:

$$\det A = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad-bc.$$

EXAMPLES

a) Use the matrix inverse to solve the system

$$\begin{cases} 2x+5y=12 \\ 3x-y=1 \end{cases}$$

Solution

$$\begin{cases} 2x+5y=12 \\ 3x-y=1 \end{cases} \iff \begin{bmatrix} 2 & 5 \\ 3 & -1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 12 \\ 1 \end{bmatrix} \iff$$

$$\iff \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 & 5 \\ 3 & -1 \end{bmatrix}^{-1}\begin{bmatrix} 12 \\ 1 \end{bmatrix} = \frac{1}{2(-1)-5\cdot 3}\begin{bmatrix} -1 & -5 \\ -3 & 2 \end{bmatrix}\begin{bmatrix} 12 \\ 1 \end{bmatrix} =$$

$$= \frac{-1}{17}\begin{bmatrix} (-1)\cdot 12+(-5)\cdot 1 \\ (-3)\cdot 12+2\cdot 1 \end{bmatrix} = \frac{-1}{17}\begin{bmatrix} -12-5 \\ -36+2 \end{bmatrix} =$$

$$= \frac{-1}{17}\begin{bmatrix} -17 \\ -34 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$ Thus solution set $S=\{(1,2)\}$

b) Similarly for the following parametric system:

$$\begin{cases} (a+1)x+(a-1)y=4a+2 \\ 2ax+(a-1)y=7a-1 \end{cases}$$

Solution

$$D=\begin{vmatrix} a+1 & a-1 \\ 2a & a-1 \end{vmatrix} = (a+1)(a-1)-2a(a-1)=(a-1)(a+1-2a)=$$

$$= (a-1)(1-a) = -(a-1)^2.$$

Case 1 : For $a\neq 1 \Rightarrow D\neq 0$, and therefore:

$$\begin{cases} (a+1)x+(a-1)y=4a+2 \\ 2ax+(a-1)y=7a-1 \end{cases} \iff \begin{bmatrix} a+1 & a-1 \\ 2a & a-1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4a+2 \\ 7a-1 \end{bmatrix} \iff$$

$$\Leftrightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a+1 & a-1 \\ 2a & a-1 \end{bmatrix}^{-1} \begin{bmatrix} 4a+2 \\ 7a-1 \end{bmatrix} =$$

$$= \frac{-1}{(a-1)^2} \begin{bmatrix} a-1 & 1-a \\ -2a & a+1 \end{bmatrix} \begin{bmatrix} 4a+2 \\ 7a-1 \end{bmatrix} =$$

$$= \frac{-1}{(a-1)^2} \begin{bmatrix} (a-1)(4a+2)+(1-a)(7a-1) \\ -2a(4a+2)+(a+1)(7a-1) \end{bmatrix} =$$

$$= \frac{-1}{(a-1)^2} \begin{bmatrix} (a-1)(4a+2-7a+1) \\ -8a^2-4a+7a^2-a+7a-1 \end{bmatrix} =$$

$$= \frac{-1}{(a-1)^2} \begin{bmatrix} (a-1)(-3a+3) \\ -a^2+2a-1 \end{bmatrix} =$$

$$= \frac{-1}{(a-1)^2} \begin{bmatrix} -3(a-1)^2 \\ -(a-1)^2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$\Leftrightarrow (x,y) = (3, 1)$. ; thus solution set $S = \{(3, 1)\}$.

<u>Case 2</u> : For $a = 1$; we have:

$$\begin{cases} (1+1)x + (1-1)y = 4\cdot 1+2 \\ 2\cdot 1x + (1-1)y = 7\cdot 1 - 1 \end{cases} \Leftrightarrow \begin{cases} 2x = 6 \\ 2x = 6 \end{cases} \Leftrightarrow x = 3$$

which gives as solution set:

$$S = \{(x,y) \in \mathbb{R}^2 \mid x = 3\} = \{(3,y) \mid y \in \mathbb{R}\}.$$

EXERCISES

18 Use the matrix inverse to solve the following systems:

a) $\begin{cases} x+3y=4 \\ 2x+y=3 \end{cases}$ b) $\begin{cases} 2x-y=3 \\ x+2y=4 \end{cases}$ c) $\begin{cases} 3x+2y=7 \\ x+3y=10 \end{cases}$

d) $\begin{cases} 2ax+(a-3)y=a-1 \\ (a-3)x+2ay=a-a^2 \end{cases}$

e) $\begin{cases} x+(a+1)y=2 \\ (a+2)x+(1-a^2)y=5 \end{cases}$ f) $\begin{cases} ax-y=1-a \\ x-ay=a-a^2 \end{cases}$

↳ Distinguish between the values of the parameter $a\in\mathbb{R}$ where the corresponding matrix is non-singular vs. singular.

19 Find all $a\in\mathbb{R}$ for which the matrix

$$A=\begin{bmatrix} a+3 & 2 \\ 1 & -a \end{bmatrix}$$

is non-singular.

20 If $A,B\in M_n(\mathbb{R})$ are non-singular, show that AB is also non-singular with the inverse given by $(AB)^{-1}=B^{-1}A^{-1}$

(21) If $A, B, G \in M_n(\mathbb{R})$ and G is non-singular, then show that

a) $GA = GB \Rightarrow A = B$

b) $AG = BG \Rightarrow A = B$

(22) If $A \in M_n(\mathbb{R})$ with $A^3 = \mathbb{0}$, show that $I - A$ is non-singular with

$$(I - A)^{-1} = I + A + A^2$$

(23) If $A \in M_n(\mathbb{R})$ satisfies $A^2 + A + I = \mathbb{0}$, show that A is non-singular and $A^{-1} = A^2$.

(24) If $A, B \in M_n(\mathbb{R})$ with A being non-singular, show that

$$(A - B) A^{-1} (A + B) = (A + B) A^{-1} (A - B)$$

(25) Let $A, B \in M_n(\mathbb{R})$ with $A \neq \mathbb{0}$ and $B \neq \mathbb{0}$. Show that:

$$AB = \mathbb{0} \Rightarrow \begin{cases} A \text{ singular} \\ B \text{ singular} \end{cases}$$

▼ Matrix Transpose

- Let $A \in M_{nm}(\mathbb{R})$ be a matrix. We define the transpose matrix $A^T \in M_{mn}(\mathbb{R})$ as:
  $\forall a \in [m] : \forall b \in [n] : (A^T)_{ab} = A_{ba}$
- Let $A \in M_n(\mathbb{R})$ be a square matrix. We say that
  A symmetric $\iff A^T = A \iff \forall a, b \in [n] : (A^T)_{ab} = A_{ba}$

► Properties

$\forall A, B \in M_{nm}(\mathbb{R}) : (A+B)^T = A^T + B^T$

$\forall \lambda \in \mathbb{R} : \forall A \in M_{nm}(\mathbb{R}) : (\lambda A)^T = \lambda A^T$

$\forall A \in M_{nk}(\mathbb{R}) : \forall B \in M_{km}(\mathbb{R}) : (AB)^T = B^T A^T$

$\forall A \in GL(n, \mathbb{R}) : (A^T)^{-1} = (A^{-1})^T$

$\forall A \in M_{nm}(\mathbb{R}) : (A^T)^T = A$

EXAMPLES

a) Prove the property
$\forall A \in M_{nk}(\mathbb{R}) : \forall B \in M_{km}(\mathbb{R}) : (AB)^T = B^T A^T$.

Solution

Let $A \in M_{nk}(\mathbb{R})$ and $B \in M_{km}(\mathbb{R})$ be given. Let $a \in [m]$ and $b \in [n]$ be given. Then:

$$[(AB)^T]_{ab} = (AB)_{ba} = \sum_{\gamma \in [k]} A_{b\gamma} B_{\gamma a} = \sum_{\gamma \in [k]} A^T_{\gamma b} B^T_{a\gamma} =$$

$$= \sum_{\gamma \in [k]} B^T_{a\gamma} A^T_{\gamma b} = (B^T A^T)_{ab}$$

It follows that

$\forall a \in [n] : \forall b \in [m] : [(AB)^T]_{ab} = (B^T A^T)_{ab}$

$\Rightarrow (AB)^T = B^T A^T$

and therefore

$\forall A \in M_{nk}(\mathbb{R}) : \forall B \in M_{km}(\mathbb{R}) : (AB)^T = B^T A^T.$ $\square$

b) Show that

$$\forall A,B \in M_n(\mathbb{R}) : \begin{cases} A,B \text{ symmetric} \\ AB = BA \end{cases} \Rightarrow AB \text{ symmetric.}$$

Solution

Let $A,B \in M_n(\mathbb{R})$ be given such that A,B symmetric and $AB = BA$. Then

$$\begin{aligned} (AB)^T &= B^T A^T && [\text{transpose of matrix product}] \\ &= BA && [\text{hypothesis: } A,B \text{ symmetric}] \\ &= AB && [\text{hypothesis: } AB = BA] \end{aligned}$$

$\Rightarrow AB$ symmetric.

It follows that

$$\forall A,B \in M_n(\mathbb{R}) : \begin{cases} A,B \text{ symmetric} \\ AB = BA \end{cases} \Rightarrow AB \text{ symmetric.}$$

EXERCISES

(26) Give proofs for all properties of the matrix transpose.

(27) Show that if $A \in M_n(\mathbb{R})$ is symmetric and non-singular, then A^{-1} is also symmetric.

(28) Let $A, B \in M_n(\mathbb{R})$. Show that
A, B, AB symmetric $\Rightarrow AB = BA$.

(29) Given $A, P \in M_n(\mathbb{R})$, show that
A symmetric $\Rightarrow B = P^T A P$ symmetric

(30) Consider the rotation matrix

$$R(\vartheta) = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}$$

a) Show that $R(\vartheta)$ is non-singular with
$[R(\vartheta)]^{-1} = R(-\vartheta)$

b) Show that
$[R(\vartheta)]^T = R(-\vartheta)$

c) For what angles $\vartheta \in \mathbb{R}$ is $R(\vartheta)$ symmetric?

(31) Let $A \in M_n(\mathbb{R})$ be a square matrix.
Show that

a) $A + A^T$ symmetric

b) $A^T A$ symmetric

(32) Let $z = a + bi \in \mathbb{C}$ with $a, b \in \mathbb{R}$ and i the imaginary unit and define

$$M(z) = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

Show that

a) $\forall z \in \mathbb{C} - \{0\} : M(1/z) = M(z)^{-1}$

b) $\forall z_1 \in \mathbb{C} : \forall z_2 \in \mathbb{C} - \{0\} : M(z_1/z_2) = M(z_1) M(z_2)^{-1}$

(33) Let $A, B \in M_n(\mathbb{R})$ be two square matrices. We say that

$$\begin{aligned} B \text{ skew-symmetric} &\Leftrightarrow \forall a, b \in [n] : B_{ab} = -B_{ba} \\ &\Leftrightarrow B^T = -B \end{aligned}$$

Show that

$$\forall A, B \in M_n(\mathbb{R}) : \begin{cases} A \text{ symmetric} \\ B \text{ skew-symmetric} \end{cases} \Rightarrow A^2 B A^2 \text{ skew-symmetric}$$

DETERMINANTS AND LINEAR SYSTEMS

Determinants

Determinants are used to find the inverse of nxn matrices, and solve nxn linear systems.

Leibnitz definition of determinants

1) Permutations

Let $[n]=\{1,2,3,\dots,n\}$. A permutation σ is a reshuffling of the order of the elements of n. Formally, σ is a bijection $\sigma:[n]\to[n]$ whereby each element of $[n]$ is mapped into a distinct element of $[n]$.

S_n = set of all permetations on $[n]$

EXAMPLE

For $n=3$:

$S_3=\{(1,2,3),(2,3,1),(3,1,2),(3,2,1),(1,3,2),(2,1,3)\}$

are the six permutations of $[3]$.

For $\sigma=(2,3,1)$: $\sigma(1)=2$, $\sigma(2)=3$, $\sigma(3)=1$.

2) Permutation parity

Let $\sigma\in S_n$ be a permutation of $[n]$. We define the parity $s(\sigma)$ of σ as:

$$\boxed{s(\sigma)=\operatorname{sign}\left[\prod_{b=1}^{n-1}\prod_{a=b+1}^{n}(\sigma(a)-\sigma(b))\right]}$$

with sign(x) defined as

$$\text{sign}(x) = \begin{cases} 1 &, \text{ if } x>0 \\ 0 &, \text{ if } x=0 \\ -1 &, \text{ if } x<0 \end{cases}$$

For $\sigma \in S_n$, $s(\sigma) = 1$ or $s(\sigma) = -1$. We say that

σ even permutation $\Leftrightarrow s(\sigma) = 1$

σ odd permutation $\Leftrightarrow s(\sigma) = -1$

EXAMPLE

For $\sigma = (3,1,4,2)$, the parity of σ is:

$$s(\sigma) = \text{sign}\left[\prod_{b=1}^{3} \prod_{a=b+1}^{4} (\sigma(a) - \sigma(b)) \right] =$$

$$= \text{sign}\left[(\sigma(2)-\sigma(1))(\sigma(3)-\sigma(1))(\sigma(4)-\sigma(1))(\sigma(3)-\sigma(2)) \times (\sigma(4)-\sigma(2))(\sigma(4)-\sigma(3)]\right]$$

$$= \text{sign}\left[(1-3)(4-3)(2-3)(4-1)(2-1)(2-4)\right]$$

$$= \text{sign}\left[(-2)(1)(-1)(3)(1)(-2)\right] = -1.$$

- A transposition is a permutation that switches only two elements of $[n]$. Every permutation can be constructed as a sequence of transpositions. An even permutation can be constructed by an even number of transpositions. An odd permutation requires an odd number of transpositions.

EXAMPLE

a) For $\sigma = (3,1,4,2)$, we can construct σ with 3 transpositions:

$$(1,2,3,4) \longrightarrow (3,2,1,4) \longrightarrow (3,2,4,1)$$

$$\longrightarrow (3,1,4,2)$$

and therefore σ is odd.

b) For $n=3$, $\mathfrak{S}_3$ has 3 even permutations and 3 odd permutations:

$A = \{\sigma \in \mathfrak{S}_3 \mid \sigma \text{ even}\}$
$= \{(1,2,3), (2,3,1), (3,1,2)\} \leftarrow$ even permutations

$B = \{\sigma \in \mathfrak{S}_3 \mid \sigma \text{ odd}\}$
$= \{(3,2,1), (1,3,2), (2,1,3)\} \leftarrow$ odd permutations

3) <u>Determinants</u>

We now use permutations to define determinants as follows:

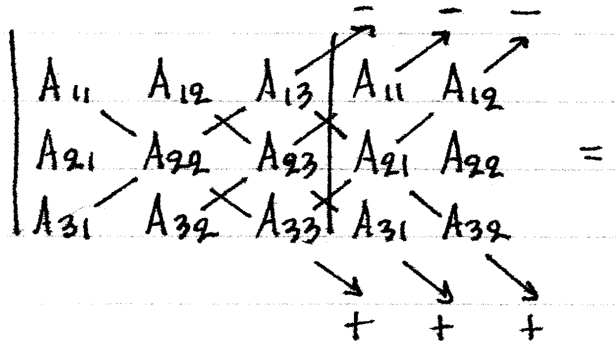
$$\boxed{\forall A \in M_n(\mathbb{R}) : \det(A) = \sum_{\sigma \in \mathfrak{S}_n} \left[s(\sigma) \prod_{a=1}^{n} A_{a,\sigma(a)} \right]}$$

1x1, 2x2, 3x3 determinants

For $n=1$: $\quad |A_{11}| = A_{11}$

For $n=2$: $\quad \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} = A_{11}A_{22} - A_{12}A_{21}$

For $n=3$: we use the Sarrus scheme:



A_{11} A_{12} A_{13} A_{11} A_{12}
A_{21} A_{22} A_{23} A_{21} A_{22}
A_{31} A_{32} A_{33} A_{31} A_{32}
− − −
+ + +
=

$$= A_{11}A_{22}A_{33} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} - A_{13}A_{22}A_{31} - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33}$$

▸ Note that there are 3 positive terms corresponding to the 3 even permutations of S_3 and 3 negative terms corresponding to the 3 odd permutations.

Fundamental properties of determinants

1) $I \in M_n(\mathbb{R})$ identity matrix $\Rightarrow \det(I) = 1$

2) $\forall A \in M_n(\mathbb{R}): \det(A^T) = \det(A)$

3) $\forall A, B \in M_n(\mathbb{R})$: $\det(AB) = \det(A)\det(B)$

4) $\forall A \in M_n(\mathbb{R})$: $(A \text{ non-singular} \Leftrightarrow \det(A) \neq 0)$

$\forall A \in M_n(\mathbb{R})$: $(A \text{ singular} \Leftrightarrow \det(A) = 0)$

↳ It follows that the set $GL(n, \mathbb{R})$ of non-singular matrices satisfies

$$GL(n, \mathbb{R}) = \{ A \in M_n(\mathbb{R}) \mid \det A \neq 0 \}$$

5) $\forall A \in GL(n, \mathbb{R})$: $\det(A^{-1}) = \dfrac{1}{\det(A)}$

↦ Determinant of lower/upper triangular matrices

- Let $A \in M_n(\mathbb{R})$ be a matrix. We say that

A lower-triangular $\Leftrightarrow \forall a, b \in [n]: (a < b \Rightarrow A_{ab} = 0)$

A upper-triangular $\Leftrightarrow \forall a, b \in [n]: (a > b \Rightarrow A_{ab} = 0)$

- It can be shown that if A is upper-triangular or lower-triangular, its determinant $\det(A)$ is given by the product of all diagonal components:

$$\boxed{\begin{array}{l} \forall A \in M_n(\mathbb{R}): \left(A \text{ lower-triangular} \Rightarrow \det A = \prod_{a=1}^{n} A_{aa}\right) \\ \forall A \in M_n(\mathbb{R}): \left(A \text{ upper-triangular} \Rightarrow \det A = \prod_{a=1}^{n} A_{aa}\right) \end{array}}$$

EXAMPLES

a) Evaluate the determinant of

$$A = \begin{bmatrix} 3 & 5 \\ 2 & 1 \end{bmatrix}$$

Solution

$$\det(A) = \begin{vmatrix} 3 & 5 \\ 2 & 1 \end{vmatrix} = 3 \cdot 1 - 5 \cdot 2 = 3 - 10 = -7.$$

b) Evaluate the determinant of

$$A = \begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix}$$, using the Sarrus rule.

Solution

$$\det(A) = \left|\begin{array}{ccc|cc} 1 & 2 & -2 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 3 & 1 & 0 & 3 \end{array}\right. =$$

$$= 1 \cdot 1 \cdot 1 + 2 \cdot 0 \cdot 0 + (-2) \cdot 1 \cdot 3 - 0 \cdot 1 \cdot (-2) - 3 \cdot 0 \cdot 1 - 1 \cdot 1 \cdot 2$$

$$= 1 + 0 - 6 - 0 - 0 - 2 = 1 - 6 - 2 = 1 - 8 = -7.$$

c) Evaluate the determinant of

$$A = \begin{bmatrix} 1 & 3 & 0 & -2 \\ 0 & -1 & 1 & 3 \\ 0 & 0 & 2 & 5 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

Solution:

A upper triangular $\Rightarrow$

$$\Rightarrow \det A = A_{11} A_{22} A_{33} A_{44}$$

$$= 1(-1) \cdot 2 \cdot 7 = -14.$$

d) Let $A, B \in M_n(\mathbb{R})$. Show that $AB = I \Rightarrow BA = I$.

Solution

Assume that $AB = I$. It follows that

$$\det(A)\det(B) = \det(AB) = \det(I) = 1 \Rightarrow$$
$$\Rightarrow \det(A)\det(B) \neq 0 \overset{*}{\Longrightarrow}$$
$$\Rightarrow \det(A) \neq 0 \land \det(B) \neq 0 \Rightarrow$$

$\Rightarrow A, B$ are non-singular

Let A^{-1}, B^{-1} be the corresponding inverse matrices. Then

$$\begin{aligned}
BA &= I(BA) && [\text{identity matrix}] \\
&= (A^{-1}A)(BA) && [A^{-1} \text{ inverse of } A] \\
&= A^{-1}[A(BA)] && [\text{associative property}] \\
&= A^{-1}[(AB)A] && [\text{associative property}] \\
&= A^{-1}IA && [\text{hypothesis } AB = I] \\
&= A^{-1}A && [\text{identity matrix}] \\
&= I && [A^{-1} \text{ inverse of } A]
\end{aligned}$$

$*$ ↳ Note that we use the contrapositive of the statement
$\forall a, b \in \mathbb{R}: (ab = 0 \Leftrightarrow (a = 0 \lor b = 0))$
which is given by
$\forall a, b \in \mathbb{R}: (ab \neq 0 \Leftrightarrow (a \neq 0 \land b \neq 0))$.

EXERCISES

① Which of the following permutations are odd and which are even? Show using both the product definition and enumeration of transpositions.

a) $\sigma = (1,3,2,4)$ c) $\sigma = (2,3,4,1)$

b) $\sigma = (3,1,4,2)$ d) $\sigma = (1,4,3,2)$

② Calculate the following determinants:

a) $\begin{vmatrix} 3 & 2 \\ 5 & 4 \end{vmatrix}$ b) $\begin{vmatrix} 2a & a+1 \\ a-1 & a \end{vmatrix}$

c) $\begin{vmatrix} 1 & 0 & 3 \\ 2 & 1 & 0 \\ 0 & 3 & 1 \end{vmatrix}$ d) $\begin{vmatrix} a & b & c \\ c & a & b \\ b & c & a \end{vmatrix}$

③ Given the matrix

$$A = \begin{bmatrix} 1 & x & x^2 \\ x^2 & 1 & x \\ x & x^2 & 1 \end{bmatrix}$$

show that

$$A \text{ singular} \Leftrightarrow x = 1$$

④ Solve with respect to x the following equations:

a) $\begin{vmatrix} 1 & 3x-4 \\ -1 & 4x+1 \end{vmatrix} = 0$ b) $\begin{vmatrix} x-1 & x^2-1 \\ 1-x^2 & x^3-1 \end{vmatrix} = 0$

⑤ Let $A, B \in M_n(\mathbb{R})$. Show that
AB singular $\Rightarrow$ (A singular $\vee$ B singular)

⑥ <u>Rotation matrix.</u>
Consider the rotation matrix

$$R(\vartheta) = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}$$

Show that $\det R(\vartheta) = 1$.

⑦ <u>Complex number matrix</u>
Let $z = a + bi \in \mathbb{C}$ be a complex number with $a, b \in \mathbb{R}$, and define

$$M(z) = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

Show that $\det M(z) = |z|, \forall z \in \mathbb{C}$.

↳ Co-factor expansion of determinants

Let $A \in M_n(\mathbb{R})$ be a square matrix. Let $a, b \in [n]$. The minor matrix $M_{ab}(A)$ is defined as the $(n-1)\times(n-1)$ square matrix obtained from A by deleting: (a) The a^{th} row of A

(b) The b^{th} column of A.

The formal definition of $M_{ab}(A)$ is given by:

$$\boxed{\forall c, d \in [n-1] : (M_{ab}(A))_{cd} = \begin{cases} A_{cd} & , \text{ if } c < a \land d < b \\ A_{c,d+1} & , \text{ if } c < a \land d \geq b \\ A_{c+1,d} & , \text{ if } c \geq a \land d < b \\ A_{c+1,d+1} & , \text{ if } c \geq a \land d \geq b \end{cases}}$$

EXAMPLE

$$\text{Given } A = \begin{bmatrix} 2 & 4 & 3 & 1 \\ 1 & 5 & 7 & 2 \\ 3 & 1 & 5 & 2 \\ 1 & 4 & 7 & 3 \end{bmatrix} \Rightarrow$$

Note that $A_{23} = 7$

$$\Rightarrow M_{23}(A) = \begin{bmatrix} 2 & 4 & 1 \\ 3 & 1 & 2 \\ 1 & 4 & 3 \end{bmatrix}$$

- We may use minor matrices to calculate determinants <u>recursively</u> as follows:

1) Row Expansion

$$\boxed{\forall a\in[n]:\ \det A=\sum_{b=1}^{n}(-1)^{a+b}A_{ab}\det(M_{ab}(A))}$$

2) Column expansion

$$\boxed{\forall b\in[n]:\ \det A=\sum_{a=1}^{n}(-1)^{a+b}A_{ab}\det(M_{ab}(A))}$$

EXAMPLE

$$\begin{vmatrix}3&1&2\\1&5&1\\2&3&1\end{vmatrix}=$$

sign of $(-1)^{a+b} \leftrightarrow \begin{bmatrix}+&-&+\\-&+&-\\+&-&+\end{bmatrix}$

$$=(-1)\cdot1\cdot\begin{vmatrix}1&2\\3&1\end{vmatrix}+(+1)\cdot5\cdot\begin{vmatrix}3&2\\2&1\end{vmatrix}+(-1)\cdot1\cdot\begin{vmatrix}3&1\\2&3\end{vmatrix}=$$

$$=-(1\cdot1-2\cdot3)+5(3\cdot1-2\cdot2)-(3\cdot3-1\cdot2)=$$

$$=-(1-6)+5(3-4)-(9-2)=$$

$$=-(-5)+5(-1)-7=5-5-7=-7.$$

▸ Method: Zero is your FRIEND.

example

$$\begin{vmatrix} 4 & 1 & 3 & 2 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 3 & 1 \end{vmatrix} = 4 \begin{vmatrix} 1 & 2 & 3 \\ 2 & 0 & 0 \\ 1 & 3 & 1 \end{vmatrix} \to =$$

$$\downarrow$$

$$= 4 \cdot (-2) \begin{vmatrix} 2 & 3 \\ 3 & 1 \end{vmatrix} = 4 \cdot (-2) \cdot (2 \cdot 1 - 3 \cdot 3)$$

$$= (-8)(2-9) = (-8)(-7) = 56$$

EXERCISE

(34) Evaluate the determinants

(a) $\begin{vmatrix} 3 & 2 & 1 & 3 \\ 0 & 1 & 9 & 2 \\ 0 & 0 & 2 & 7 \\ 0 & 0 & 0 & 5 \end{vmatrix}$ (b) $\begin{vmatrix} 1 & 2 & 1 & 5 \\ 3 & 0 & 0 & 0 \\ 4 & 0 & 3 & 7 \\ 2 & 0 & 2 & 1 \end{vmatrix}$

(c) $\begin{vmatrix} 1 & 3 & 2 & 7 & 5 \\ 5 & 0 & 7 & 0 & 0 \\ 2 & 0 & 2 & 3 & 1 \\ 3 & 0 & 0 & 0 & 0 \\ 8 & 0 & 1 & 4 & 2 \end{vmatrix}$

↳ Simplification of determinants

The calculation of determinants can be simplified considerably by using the following properties:

1) If we transpose 2 rows or 2 columns, the determinant changes sign.

$$\text{e.g.} \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} \begin{array}{l} \leftarrow \\ \\ \leftarrow \end{array} = - \begin{vmatrix} c_1 & c_2 & c_3 \\ b_1 & b_2 & b_3 \\ a_1 & a_2 & a_3 \end{vmatrix}$$

2) If 2 rows or 2 columns are identical, then the determinant is equal to 0.

$$\text{e.g.} \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ a_1 & a_2 & a_3 \end{vmatrix} = 0$$

3) If we multiply a row or column by $\lambda \in \mathbb{R}$, then the determinant itself is multiplied by λ.

$$\text{e.g.} \ \lambda \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = \begin{vmatrix} a_1 & \lambda a_2 & a_3 \\ b_1 & \lambda b_2 & b_3 \\ c_1 & \lambda c_2 & c_3 \end{vmatrix} = \begin{vmatrix} \lambda a_1 & \lambda a_2 & \lambda a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}$$

It follows that

- 1 We can pull out a common factor from any row or column.

e.g. $$\underset{\uparrow}{\begin{vmatrix}3&2&7\\5&4&-1\\8&-8&12\end{vmatrix}} = 2\begin{vmatrix}3&1&7\\5&2&-1\\8&-4&12\end{vmatrix}\leftarrow = 2\cdot4\cdot\begin{vmatrix}3&1&7\\5&2&-1\\2&-1&3\end{vmatrix}$$

- 2 If all the elements of a row or column are 0, then the determinant is 0.

e.g. $$\begin{vmatrix}2&0&3\\7&0&2\\1&0&4\end{vmatrix} = 0$$

- 3 If $A \in M_{nn}(\mathbb{R}) \Rightarrow \det(\lambda A) = \lambda^n \det(A), \forall \lambda \in \mathbb{R}$.

4) When every element of a row or column is written as a sum of two numbers, then the determinant can be rewritten as the sum of two determinants.

e.g. $$\begin{vmatrix}a_1 & a_2 & a_3\\ b_1+c_1 & b_2+c_2 & b_3+c_3\\ d_1 & d_2 & d_3\end{vmatrix} = \begin{vmatrix}a_1&a_2&a_3\\b_1&b_2&b_3\\d_1&d_2&d_3\end{vmatrix} + \begin{vmatrix}a_1&a_2&a_3\\c_1&c_2&c_3\\d_1&d_2&d_3\end{vmatrix}$$

5) If we add to the elements of a row (or column) the elements of another row (or column) multiplied by a common factor λ, then the value of the determinant does not change.

e.g.
$$\begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} \begin{array}{l} \cdot\lambda \\ \hookleftarrow \\ \end{array} = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1+\lambda a_1 & b_2+\lambda a_2 & b_3+\lambda a_3 \\ c_1 & c_2 & c_3 \end{vmatrix}$$

6) When the elements above OR bellow the diagonal are all 0, then the determinant is equal to the product of the diagonal elements.

e.g.
$$\begin{vmatrix} a_1 & a_2 & a_3 \\ 0 & b_2 & b_3 \\ 0 & 0 & c_3 \end{vmatrix} = a_1 b_2 c_3$$

So, to simplify a determinant

- •1 Check if common factors can be pulled out via (3)
- •2 Use (1), (5) to diagonalize the determinant (i.e. create ZEROES!) so you can then use (6)
- •3 If you run into identical rows or columns then use (2).

EXAMPLES

a) Evaluate the determinant

$$\begin{vmatrix}1&2&-1&2\\2&-4&-3&3\\0&4&0&1\\1&6&0&1\end{vmatrix}$$

Solution

$$\begin{vmatrix}1&2&-1&2\\2&-4&-3&3\\0&4&0&1\\1&6&0&1\end{vmatrix}\begin{matrix}\cdot(-2)\quad\cdot(-1)\\ \hookleftarrow\qquad\quad\\ \\ \longleftarrow\!\!\lrcorner\end{matrix} =$$

$$= \begin{vmatrix}1&2&-1&2\\2+(-2)\cdot1&-4+(-2)\cdot2&-3+(-2)(-1)&3+(-2)\cdot2\\0&4&0&1\\1+(-1)\cdot1&6+(-1)\cdot2&0+(-1)(-1)&1+(-1)2\end{vmatrix} =$$

$$= \begin{vmatrix}1&2&-1&2\\0&-8&-1&-1\\0&4&0&1\\0&4&1&-1\end{vmatrix} = (+1)\cdot1\cdot\begin{vmatrix}-8&-1&-1\\4&0&1\\4&1&-1\end{vmatrix} =$$

$$= 4\begin{vmatrix} -2 & -1 & -1 \\ 1 & 0 & 1 \\ 1 & 1 & -1 \end{vmatrix}\begin{matrix} \cdot 1 \\ \big\downarrow \\ \leftarrow \end{matrix} = 4\begin{vmatrix} -2 & -1 & -1 \\ 1 & 0 & 1 \\ 1-2 & 1-1 & -1-1 \end{vmatrix} =$$

$$= 4\begin{vmatrix} \boxed{-2} & \boxed{-1} & \boxed{-1} \\ 1 & \boxed{0} & 1 \\ -1 & \boxed{0} & -2 \end{vmatrix} = 4\cdot(-1)(-1)\begin{vmatrix} 1 & 1 \\ -1 & -2 \end{vmatrix} =$$

$$= 4\left(1(-2) - 1(-1)\right) = 4(-2+1) = -4$$

↳ We use determinant propertries to zero out a column or a row. Then we perform a co-factor expansion. This reduces the size of the determinant. We repeat, all the way down to 2x2 size.

8) Show that

$$\begin{vmatrix} a+b & b+c & c+a \\ c+a & a+b & b+c \\ b+c & c+a & a+b \end{vmatrix} = 2(a+b+c)(a^2+b^2+c^2-ab-bc-ca)$$

Solution

$$\begin{vmatrix} a+b & b+c & c+a \\ c+a & a+b & b+c \\ b+c & c+a & a+b \end{vmatrix} = \begin{vmatrix} (a+b)+(b+c)+(c+a) & b+c & c+a \\ (c+a)+(a+b)+(b+c) & a+b & b+c \\ (b+c)+(c+a)+(a+b) & c+a & a+b \end{vmatrix} =$$

$\uparrow$ —— $\cdot 1$

$\uparrow$ ———— $\cdot 1$

$$= \begin{vmatrix} 2(a+b+c) & b+c & c+a \\ 2(a+b+c) & a+b & b+c \\ 2(a+b+c) & c+a & a+b \end{vmatrix} = 2(a+b+c)\begin{vmatrix} 1 & b+c & c+a \\ 1 & a+b & b+c \\ 1 & c+a & a+b \end{vmatrix}\begin{array}{l} (-1) \\ \leftarrow \\ \leftarrow \end{array}$$

$$= 2(a+b+c)\begin{vmatrix} 1 & b+c & c+a \\ 0 & (a+b)-(b+c) & (b+c)-(c+a) \\ 0 & (c+a)-(b+c) & (a+b)-(c+a) \end{vmatrix}$$

$$= 2(a+b+c)\begin{vmatrix} 1 & b+c & c+a \\ 0 & a-c & b-a \\ 0 & a-b & b-c \end{vmatrix} = 2(a+b+c)\cdot(+1)(1)\begin{vmatrix} a-c & b-a \\ a-b & b-c \end{vmatrix}$$

$$= 2(a+b+c)\left[(a-c)(b-c)-(b-a)(a-b)\right]$$

$$= 2(a+b+c)\left[(a-c)(b-c)+(a-b)^2\right] =$$
$$= 2(a+b+c)(\underline{ab}-ac-bc+c^2+a^2-\underline{2ab}+b^2)$$
$$= 2(a+b+c)(a^2+b^2+c^2-ab-bc-ca).$$

EXERCISES

⑨ Evaluate the following determinants:

a) $\begin{vmatrix} 3 & 5 & 8 \\ 3 & 6 & 9 \\ 3 & 7 & 4 \end{vmatrix}$ b) $\begin{vmatrix} x & x+1 & x+3 \\ y & y+1 & y+3 \\ 1 & 1 & 1 \end{vmatrix}$

c) $\begin{vmatrix} 13 & 16 & 19 \\ 14 & 17 & 20 \\ 15 & 18 & 21 \end{vmatrix}$ d) $\begin{vmatrix} 1 & -2 & 0 & 3 \\ 1 & 1 & 2 & 1 \\ 3 & 1 & -1 & 4 \\ 5 & 1 & 2 & -1 \end{vmatrix}$

e) $\begin{vmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{vmatrix}$ f) $\begin{vmatrix} -1 & 0 & 2 & 1 & -3 \\ 1 & 2 & 3 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 4 & 5 & 1 & -1 \\ 0 & 1 & 2 & 0 & -2 \end{vmatrix}$

⑩ Solve the following equations

a) $\begin{vmatrix} x-3 & 4 & x \\ 3x-2 & -6 & 2x-1 \\ 4x-3 & 2 & x^2-3 \end{vmatrix} = 0$ b) $\begin{vmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & x \\ 1 & 2 & 1 & x^2 \\ 1 & 3 & 3 & x^3 \end{vmatrix} = 0$

11 Show that

a) $$\begin{vmatrix} 1 & a & b+c \\ 1 & b & c+a \\ 1 & c & a+b \end{vmatrix} = 0$$

b) $$\begin{vmatrix} 1 & 1 & 1 \\ a & b & c \\ a^2 & b^2 & c^2 \end{vmatrix} = (b-c)(c-a)(a-b)$$

c) $$\begin{vmatrix} a-b-c & 2a & 2a \\ 2b & b-c-a & 2b \\ 2c & 2c & c-a-b \end{vmatrix} = (a+b+c)^3$$

d) $$\begin{vmatrix} x+y & z & z \\ y & z+x & y \\ x & x & z+y \end{vmatrix} = 4xyz$$

e) $$\begin{vmatrix} a & b & c \\ a^2 & b^2 & c^2 \\ a^3 & b^3 & c^3 \end{vmatrix} = abc(a-b)(b-c)(c-a)$$

f) $$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1+a & 1 \\ 1 & 1 & 1+b \end{vmatrix} = ab$$

g) $\begin{vmatrix} 1 & -c & b \\ c & 1 & -a \\ -b & a & 1 \end{vmatrix} = a^2+b^2+c^2+1$

h) $\begin{vmatrix} a & a & a & a \\ a & b & b & b \\ a & b & c & c \\ a & b & c & d \end{vmatrix} = a(b-a)(c-b)(d-c)$

i) $\begin{vmatrix} a^2 & (a+1)^2 & (a+2)^2 & (a+3)^2 \\ b^2 & (b+1)^2 & (b+2)^2 & (b+3)^2 \\ c^2 & (c+1)^2 & (c+2)^2 & (c+3)^2 \\ d^2 & (d+1)^2 & (d+2)^2 & (d+3)^2 \end{vmatrix} = 0$

j) $\begin{vmatrix} 1 & a & a^2 & a^3+bcd \\ 1 & b & b^2 & b^3+cda \\ 1 & c & c^2 & c^3+dab \\ 1 & d & d^2 & d^3+abc \end{vmatrix} = 0$

k) $\begin{vmatrix} 1 & a & a^2 & a^3 & a^4 \\ a^4 & 1 & a & a^2 & a^3 \\ a^3 & a^4 & 1 & a & a^2 \\ a^2 & a^3 & a^4 & 1 & a \\ a & a^2 & a^3 & a^4 & 1 \end{vmatrix} = (1-a^5)^4$

l) $$\begin{vmatrix} 1 & 1 & 1 \\ a^2 & b^2 & c^2 \\ a^3 & b^3 & c^3 \end{vmatrix} = (ab+bc+ca)\begin{vmatrix} 1 & 1 & 1 \\ a & b & c \\ a^2 & b^2 & c^2 \end{vmatrix}$$

m) $$\begin{vmatrix} a_1 & b_1 & a_1x^2+b_1x+c_1 \\ a_2 & b_2 & a_2x^2+b_2x+c_2 \\ a_3 & b_3 & a_3x^2+b_3x+c_3 \end{vmatrix} = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

n) $$\begin{vmatrix} 1 & a & b & 1 \\ 1 & a & a & a \\ a & 1 & ab & b \\ a & a & ab & 1 \end{vmatrix} = (a-b)(a-1)(1-ab)$$

o) $$\begin{vmatrix} a & -b & -a & b \\ b & a & -b & -a \\ c & -d & c & -d \\ d & c & d & c \end{vmatrix} = 4(a^2+b^2)(c^2+d^2)$$

p) $$\begin{vmatrix} a & b & b & b \\ a & b & a & a \\ b & b & a & b \\ a & a & a & b \end{vmatrix} = -(a-b)^4$$

▼ Matrix inverse, in general

Recall that for a 2×2 matrix

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

we have

$$ad-bc \neq 0 \Rightarrow A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

For larger matrices, we use the following theory:

Def : Let $A \in M_n(\mathbb{R})$ be a square matrix. We define the adjugate matrix $\mathrm{adj}(A)$ such that

$$\forall a,b \in [n] : [\mathrm{adj}(A)]_{ab} = (-1)^{a+b} \det(M_{ba}(A))$$

Thm : Let $A \in GL(n,\mathbb{R})$ be a non-singular square matrix. Then

$$A^{-1} = \left(\frac{1}{\det(A)}\right)\mathrm{adj}(A)$$

EXAMPLE

Find the matrix inverse of $A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & -1 \\ 1 & 2 & 5 \end{bmatrix}$

Solution

Since,

$$\det(A) = \begin{vmatrix} 1 & 0 & -1 \\ 2 & 1 & -1 \\ 1 & 2 & 5 \end{vmatrix} \begin{array}{l} (-2) \quad (-1) \\ \hookleftarrow \\ \longleftarrow \end{array} = \begin{vmatrix} 1 & 0 & -1 \\ 0 & 1+(-2)\cdot 0 & -1+(-2)(-1) \\ 0 & 2+(-1)\cdot 0 & 5+(-1)(-1) \end{vmatrix}$$

$$= \begin{vmatrix} \boxed{1} & \boxed{0 \quad -1} \\ \boxed{\begin{matrix} 0 \\ 0 \end{matrix}} & \begin{matrix} 1 & 1 \\ 2 & 6 \end{matrix} \end{vmatrix} = (+1)\cdot 1 \cdot \begin{vmatrix} 1 & 1 \\ 2 & 6 \end{vmatrix} = 1\cdot 6 - 1\cdot 2 = 4$$

and

$$[\operatorname{adj}(A)]_{11} = (-1)^{1+1} \det(M_{11}(A)) = \begin{vmatrix} 1 & -1 \\ 2 & 5 \end{vmatrix} = 1\cdot 5 - (-1)\cdot 2$$

$$= 5+2 = 7$$

$$[\operatorname{adj}(A)]_{12} = (-1)^{1+2} \det(M_{21}(A)) = - \begin{vmatrix} 0 & -1 \\ 2 & 5 \end{vmatrix} =$$

$$= -(0\cdot 5 - (-1)\cdot 2) = -(0+2) = -2$$

$$[\operatorname{adj}(A)]_{13} = (-1)^{1+3} \det(M_{31}(A)) = \begin{vmatrix} 0 & -1 \\ 1 & -1 \end{vmatrix} =$$

$$= 0(-1) - (-1)\cdot 1 = 1$$

$$[\mathrm{adj}(A)]_{21} = (-1)^{2+1}\det(M_{12}(A)) = -\begin{vmatrix} 2 & -1 \\ 1 & 5 \end{vmatrix} =$$

$$= -(2\cdot 5 - (-1)1) = -(10+1) = -11$$

$$[\mathrm{adj}(A)]_{22} = (-1)^{2+2}\det(M_{22}(A)) = \begin{vmatrix} 1 & -1 \\ 1 & 5 \end{vmatrix} =$$

$$= 1\cdot 5 - (-1)\cdot 1 = 5+1 = 6$$

$$[\mathrm{adj}(A)]_{23} = (-1)^{2+3}\det(M_{32}(A)) = -\begin{vmatrix} 1 & -1 \\ 2 & -1 \end{vmatrix} =$$

$$= -[1(-1) - (-1)2] = -(-1+2) = -1$$

$$[\mathrm{adj}(A)]_{31} = (-1)^{3+1}\det(M_{13}(A)) = \begin{vmatrix} 2 & 1 \\ 1 & 2 \end{vmatrix} = 2\cdot 2 - 1\cdot 1 = 3$$

$$[\mathrm{adj}(A)]_{32} = (-1)^{3+2}\det(M_{23}(A)) = -\begin{vmatrix} 1 & 0 \\ 1 & 2 \end{vmatrix} =$$

$$= -(1\cdot 2 - 0\cdot 1) = -2$$

$$[\mathrm{adj}(A)]_{33} = (-1)^{3+3}\det(M_{33}(A)) = \begin{vmatrix} 1 & 0 \\ 2 & 1 \end{vmatrix} = 1\cdot 1 - 2\cdot 0 = 1$$

it follows that

$$\mathrm{adj}(A) = \begin{bmatrix} 7 & -2 & 1 \\ -11 & 6 & -1 \\ 3 & -2 & 1 \end{bmatrix} \Rightarrow$$

$$\Rightarrow A^{-1} = \frac{1}{\det A}\,\mathrm{adj}(A) = \frac{1}{4}\begin{bmatrix} 7 & -2 & 1 \\ -4 & 6 & -1 \\ 3 & -2 & 1 \end{bmatrix}$$

EXERCISES

(12) Find the inverse matrix A^{-1} for the following matrices

a) $A = \begin{bmatrix} 2 & 3 & -1 \\ 0 & 1 & 1 \\ 1 & 2 & 5 \end{bmatrix}$ b) $A = \begin{bmatrix} 1 & -5 & 0 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix}$

(13) If $(2I-A)^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix}$, evaluate the matrix A.

(14) If $A^{-1}B^{-1} = \begin{bmatrix} 5 & 0 \\ 2 & -1 \end{bmatrix}$ and $A = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}$ then evaluate the matrix B.

(15) If $A^{-1}B^{-1} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 7 \\ -2 & -4 & -5 \end{bmatrix}$, then evaluate the matrix BA.

(16) Solve for the matrix $X \in M_3(\mathbb{R})$:

$$\begin{bmatrix} 1 & -2 & 3 \\ 4 & 1 & 5 \\ 5 & 0 & 8 \end{bmatrix} X = 7 \begin{bmatrix} 1 & -1 & 2 \\ 0 & 1 & 4 \\ 1 & 1 & 3 \end{bmatrix}$$

▼ $n\times n$ linear system of equations

- An $n\times n$ linear system of equations is a system of the form:

$$\begin{cases} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n = b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n = b_2 \\ \vdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nn}x_n = b_n \end{cases} \qquad (1)$$

- This system can be rewritten as a matrix equation:

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

or equivalently as

$$Ax = b \qquad (2)$$

with $A \in M_n(\mathbb{R})$ and $x, b \in M_{n1}(\mathbb{R})$.

- We say that

  Eq.(2) is homogeneous $\Leftrightarrow$ $b = \mathbf{0}$

  Eq.(2) is inhomogeneous $\Leftrightarrow$ $b \neq \mathbf{0}$

- Solution techniques for solving linear systems include:

a) Matrix inverse method
b) Cramer's rule
c) Gaussian elimination

Matrix inverse method

We have already explained that if $\det A \neq 0$, then any linear system can be solved via the property

$$\forall A \in GL(n,\mathbb{R}) : \forall x, b \in M_{n1}(\mathbb{R}) : (Ax = b \Leftrightarrow x = A^{-1}b)$$

However, due to the difficulty of calculating A^{-1}, this method is recommended only for 2×2 systems, as was explained earlier.

Cramer's rule

We write

$$A = [A_1 \; A_2 \; A_3 \cdots A_n]$$

with $A_1, A_2, A_3, \ldots, A_n \in M_{n1}(\mathbb{R})$ the columns of the matrix A, and define the determinants

$$D = \det A = \det([A_1 \; A_2 \; A_3 \cdots A_n])$$
$$D_1 = \det([b \; A_2 \; A_3 \cdots A_n])$$
$$D_2 = \det([A_1 \; b \; A_3 \cdots A_n])$$

$$D_3 = \det([A_1\ \ A_2\ \ b \cdots A_n])$$

$$\vdots$$

$$D_n = \det([A_1\ \ A_2\ \ A_3 \cdots\ \ b])$$

Note that for any $k \in [n]$, in the determinant D_k, we replace the k^{th} column of A with the column matrix b.

- Cramer's method for solving linear systems is based on the following theorem:

Thm: Given the linear system $Ax=b$ with $A \in M_n(\mathbb{R})$ and $x, b \in M_{n1}(\mathbb{R})$.

a) If $D \neq 0$, then $Ax=b$ has a unique solution given by $\forall k \in [n]: x_k = D_k / D$.

b) $\begin{cases} D = 0 \\ \exists k \in [n]: D_k \neq 0 \end{cases} \Rightarrow Ax=b$ has no solutions

Remark: Note that the theorem is inconclusive when

$$\begin{cases} D = 0 \\ \forall k \in [n]: D = 0 \end{cases}$$

Then, the system needs to be investigated via Gaussian Elimination method.

a) $\begin{cases} \lambda x + (\lambda-2)y = \lambda+1 \\ (\lambda+1)x - (\lambda-2)y = \lambda \end{cases}$ (1)

Solution

$$D = \begin{vmatrix} \lambda & \lambda-2 \\ \lambda+1 & -(\lambda-2) \end{vmatrix} = (\lambda-2)\begin{vmatrix} \lambda & 1 \\ \lambda+1 & -1 \end{vmatrix} = (\lambda-2)[\lambda(-1) - (\lambda+1)\cdot 1]$$

$$= (\lambda-2)(-\lambda-\lambda-1) = -(\lambda-2)(2\lambda+1)$$

$$D_x = \begin{vmatrix} \lambda+1 & \lambda-2 \\ \lambda & -(\lambda-2) \end{vmatrix} = (\lambda-2)\begin{vmatrix} \lambda+1 & 1 \\ \lambda & -1 \end{vmatrix} = (\lambda-2)[(\lambda+1)(-1) - 1\cdot\lambda].$$

$$= (\lambda-2)(-\lambda-1-\lambda) = -(\lambda-2)(2\lambda+1)$$

$$D_y = \begin{vmatrix} \lambda & \lambda+1 \\ \lambda+1 & \lambda \end{vmatrix} = \lambda^2 - (\lambda+1)^2 = (\lambda-(\lambda+1))(\lambda+(\lambda+1)) =$$

$$= (\lambda-\lambda-1)(\lambda+\lambda+1) = -(2\lambda+1)$$

Note that $D = 0 \Leftrightarrow -(\lambda-2)(2\lambda+1) = 0 \Leftrightarrow$

$\Leftrightarrow \lambda-2 = 0 \vee 2\lambda+1 = 0 \Leftrightarrow \lambda = 2 \vee \lambda = -1/2 \Leftrightarrow$

$\Leftrightarrow \lambda \in \{2, -1/2\}$.

Case 1: If $\lambda \in \mathbb{R} - \{-1/2, 2\}$, then the system has a unique solution given by:

$$x = \frac{D_x}{D} = \frac{-(\lambda-2)(2\lambda+1)}{-(\lambda-2)(2\lambda+1)} = 1$$

$$y = \frac{D_y}{D} = \frac{-(2\lambda+1)}{-(\lambda-2)(2\lambda+1)} = \frac{1}{\lambda-2}$$

Case 2: If $\lambda = 2$, then

$D = 0 \wedge D_x = 0 \wedge D_y = -5 \neq 0 \Rightarrow$

$\Rightarrow$ the system is inconsistent (i.e. no solutions).

<u>Case 3</u> : If $\lambda = -1/2$, then
$$D = 0 \land D_x = 0 \land D_y = 0$$
so we have to solve the system explicitly:

$$(1) \Leftrightarrow \begin{cases} (-1/2)x + (-1/2-2)y = -1/2+1 \\ (-1/2+1)x - (-1/2-2)y = -1/2 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} -x + (-1-4)y = -1+2 \\ (-1+2)x - (-1-4)y = -1 \end{cases} \Leftrightarrow \begin{cases} -x - 5y = 1 \\ x + 5y = -1 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow x + 5y = 1 \Leftrightarrow x = 1 - 5y \Leftrightarrow (x,y) = (1-5y, y)$$

$$\Leftrightarrow (x,y) \in \{(1-5y, y) \mid y \in \mathbb{R}\}.$$

To summarize, the solution set is:

$$S = \begin{cases} \{(1, 1/(\lambda-2))\} & , \text{ if } \lambda \in \mathbb{R} - \{-1/2, 2\} \\ \varnothing & , \text{ if } \lambda = -1/2 \\ \{(1-5y, y) \mid y \in \mathbb{R}\} & , \text{ if } \lambda = 2. \end{cases}$$

EXERCISES

⑰ Use Cramer's rule to solve the following linear systems:

a) $\begin{cases} x-y=0 \\ 3x+2y=5 \end{cases}$

(1,1)

b) $\begin{cases} x+2y-z=0 \\ 2x-y+3z=0 \\ x+y+z=2 \end{cases}$

(−2,2,2)

c) $\begin{cases} x-y+z=3 \\ 2x+y-3z=10 \\ x+5y-9z=8 \end{cases}$

(no solutions)

d) $\begin{cases} 2x-y-z-w=-1 \\ x-2y+z+w=-2 \\ x+y-2z+w=4 \\ x+y+z-2w=-8 \end{cases}$

(−2,−1,−3,1)

e) $\begin{cases} x+y+z+w=2 \\ 2x-w+3z=9 \\ -x+2y-z+2w=-5 \\ 3x+y-w=4 \end{cases}$

$(x,y,z,w)=(1,0,2,-1)$

⑱ Solve the following linear systems in terms of the parameter $a \in \mathbb{R}$.

a) $\begin{cases} ax+y+z=1 \\ x+ay+z=a \\ x+y+az=a^2 \end{cases}$

b) $\begin{cases} x-ay+z=a \\ x+y+z=-1 \\ ax+y-a^2z=1. \end{cases}$

c) $\begin{cases} x+y+z=1 \\ ax+by+cz=d \\ a^2x+b^2y+c^2z=d^2 \end{cases}$

d) $\begin{cases} x+y+z=1+c \\ x+(1+a)y+z=1 \\ x+y+(1+b)z=1 \end{cases}$

Method of Gaussian elimination

- We represent the linear system $Ax=b$ in terms of an <u>augmented matrix</u> M:

$$\left[\begin{array}{cccc|c} A_{11} & A_{12} & \cdots & A_{1n} & b_1 \\ A_{21} & A_{22} & \cdots & A_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} & b_m \end{array}\right] \longleftarrow \text{each row represents an equation}$$

- We say that two augmented matrices M_1 and M_2 are equivalent (notation: $M_1 \sim M_2$) if and only if the corresponding linear systems have the same solution set.

Properties

- The following transformations map an augmented matrix M_1 to an equivalent augmented matrix M_2.

1) <u>Transposition</u>: We can swap any two rows (but not columns).

e.g. $$\left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{array}\right] \begin{array}{c} \leftarrow \\ \\ \leftarrow \end{array} \sim \left[\begin{array}{ccc|c} a_3 & b_3 & c_3 & d_3 \\ a_2 & b_2 & c_2 & d_2 \\ a_1 & b_1 & c_1 & d_1 \end{array}\right]$$

2) <u>Scalar Multiplication</u>: We can multiply any row (but not a column) with a non-zero scalar $\lambda \in \mathbb{R}-\{0\}$.

e.g. $$\left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{array}\right] \begin{array}{c} \\ \cdot\lambda \\ \\ \end{array} \sim \left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ \lambda a_2 & \lambda b_2 & \lambda c_2 & \lambda d_2 \\ a_3 & b_3 & c_3 & d_3 \end{array}\right]$$

3) <u>Linear combination</u> : We can add to any row (but not a column) any other row multiplied by a non-zero scalar $\lambda \in \mathbb{R}-\{0\}$

e.g. $$\left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{array}\right] \begin{array}{c} \cdot\lambda \\ \hookleftarrow \\ \\ \end{array} \sim \left[\begin{array}{ccc|c} a_1 & b_1 & c_1 & d_1 \\ a_2+\lambda a_1 & b_2+\lambda b_1 & c_2+\lambda c_1 & d_2+\lambda d_1 \\ a_3 & b_3 & c_3 & d_3 \end{array}\right]$$

- Note that these properties are somewhat different from the corresponding properties of determinants.

▶ <u>Method</u>

- Using these properties, we try to diagonalize the augmented matrix defering fractional arithmetic as much as possible to the very last step. We work on the augmented matrix one column at a time.
- If during this process we get a row of the form

$$0 \quad 0 \quad 0 \ldots 0 \,|\, a$$

then: a) If $a \neq 0$, the system is inconsistent and we stop work.

b) If $a=0$, then the row corresponds to an identity and may then be deleted from the augmented matrix.

EXAMPLES

a) $\begin{cases} 2x-y=1 \\ x+y=3 \\ 3x+y=0 \end{cases}$ ⟵● An overdetermined system: more equations than unknowns

Solution

$$M=\left[\begin{array}{cc|c} 2 & -1 & 1 \\ 1 & 1 & 3 \\ 3 & 1 & 0 \end{array}\right] \begin{matrix} \circlearrowleft \\ \leftarrow \\ \end{matrix} \sim \left[\begin{array}{cc|c} 1 & 1 & 3 \\ 2 & -1 & 1 \\ 3 & 1 & 0 \end{array}\right] \begin{matrix} (-2) \quad (-3) \\ \leftarrow \\ \leftarrow \end{matrix} \sim$$

$$\sim \left[\begin{array}{cc|c} 1 & 1 & 3 \\ 0 & -1+(-2)\cdot 1 & 1+(-2)3 \\ 0 & 1+(-3)\cdot 1 & 0+(-3)3 \end{array}\right] \sim$$

$$\sim \left[\begin{array}{cc|c} 1 & 1 & 3 \\ 0 & -3 & -5 \\ 0 & -2 & -9 \end{array}\right] \begin{matrix} \\ \cdot 2 \\ \cdot(-3) \end{matrix} \sim \left[\begin{array}{cc|c} 1 & 1 & 3 \\ 0 & -6 & -10 \\ 0 & 6 & 27 \end{array}\right] \begin{matrix} \\ \cdot 1 \\ \hookleftarrow \end{matrix} \sim$$

$$\sim \left[\begin{array}{cc|c} 1 & 1 & 3 \\ 0 & -6 & -10 \\ 0 & 0 & 27-10 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & 1 & 3 \\ 0 & -6 & -10 \\ 0 & 0 & 17 \end{array}\right]$$

therefore the system has no solutions.

b) $\begin{cases} x+z+4w+2v=3 \\ y+2w-V=-1 \\ -x+3y+2z=-2 \end{cases}$ ⟵● An underdetermined system: less equations than unknowns.

Solution

Since,

$$\begin{cases} x+z+4w+2v=3\\ y+2w-v=-1\\ -x+3y+2z=-2\end{cases} \Leftrightarrow \begin{cases}1x+0y+1z+4w+2v=3\\ 0x+1y+0z+2w-v=-1\\ -1x+3y+2z+0w+0v=-2\end{cases} \qquad (1)$$

the corresponding augmented matrix is:

$$M=\left[\begin{array}{ccccc|c} 1 & 0 & 1 & 4 & 2 & 3\\ 0 & 1 & 0 & 2 & -1 & -1\\ -1 & 3 & 2 & 0 & 0 & -2\end{array}\right]\begin{array}{l}\cdot 1\\ \\ \end{array}$$

$$\sim\left[\begin{array}{ccccc|c} 1 & 0 & 1 & 4 & 2 & 3\\ 0 & 1 & 0 & 2 & -1 & -1\\ 0 & 3+0 & 2+1 & 0+4 & 0+2 & -2+3\end{array}\right]$$

$$\sim\left[\begin{array}{ccccc|c} 1 & 0 & 1 & 4 & 2 & 3\\ 0 & 1 & 0 & 2 & -1 & -1\\ 0 & 3 & 3 & 4 & 2 & 1\end{array}\right]\begin{array}{l} \\ (-3)\\ \\ \end{array}$$

$$\sim\left[\begin{array}{ccccc|c} 1 & 0 & 1 & 4 & 2 & 3\\ 0 & 1 & 0 & 2 & -1 & -1\\ 0 & 0 & 3+(-3)\cdot 0 & 4+(-3)2 & 2+(-3)(-1) & 1+(-1)(-3)\end{array}\right]$$

$$\sim\left[\begin{array}{ccccc|c} 1 & 0 & 1 & 4 & 2 & 3\\ 0 & 1 & 0 & 2 & -1 & -1\\ 0 & 0 & 3 & -2 & 5 & 4\end{array}\right]\begin{array}{l}\cdot 3\\ \\ \\ \end{array}\sim$$

$$\sim\left[\begin{array}{ccccc|c} 3 & 0 & 3 & 12 & 6 & 9\\ 0 & 1 & 0 & 2 & -1 & -1\\ 0 & 0 & 3 & -2 & 5 & 4\end{array}\right]\begin{array}{l} \\ \\ (-1)\end{array}\sim$$

$$\sim \left[\begin{array}{ccccc|c} 3 & 0 & 0 & 12+(-1)(-2) & 6+(-1)5 & 9+(-1)4 \\ 0 & 1 & 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -2 & 5 & 4 \end{array}\right]$$

$$\sim \left[\begin{array}{ccccc|c} 3 & 0 & 0 & 14 & 1 & 5 \\ 0 & 1 & 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -2 & 5 & 4 \end{array}\right]$$

and it follows that

$$\text{Eq.}(1) \iff \begin{cases} 3x+14w+v=5 \\ y+2w-v=-1 \\ 3z-2w+5v=4 \end{cases} \iff \begin{cases} 3x=5-14w-v \\ y=-1-2w+v \\ 3z=4+2w-5v \end{cases}$$

$$\iff \begin{cases} x=(5/3)-(14/3)w-(1/3)v \\ y=-1-2w+v \\ z=(4/3)+(2/3)w-(5/3)v \end{cases}$$

$$\iff (x,y,z)=((5/3)-(14/3)w-(1/3)v,$$
$$-1-2w+v,\ (4/3)+(2/3)w-(5/3)v)$$
$$=(5/3,-1,4/3)+w(-14/3,-2,2/3)+$$
$$+v(-1/3,1,-5/3)$$

$$\iff (x,y,z,w,v)=(5/3,-1,4/3,0,0)+w(-14/3,-2,2/3,1,0)$$
$$+v(-1/3,1,-5/3,0,1)$$

EXERCISES

(19) Solve the following systems using Gaussian Elimination

a) $\begin{cases} x-2y=-4 \\ 3x+y=9 \\ x+5y=17 \end{cases}$ $(2,3)$

b) $\begin{cases} x+\ z=4 \\ 2x-y+3z=9 \\ 2y-z=1 \\ 3x+y-2z=-1 \end{cases}$ $(1,2,3)$

c) $\begin{cases} x-y-2z=6 \\ 3x-3y-6z=1 \end{cases}$ (inconsistent)

d) $\begin{cases} x+y+w=4 \\ y+w+z=-2 \\ x+w+z=1 \end{cases}$ $(z+6, z+3, -2z-5, z)$

e) $\begin{cases} x+2y+4z=0 \\ y-2z=0 \\ x+8z=0 \end{cases}$ $(-8z, 2z, z)$

GRAPH THEORY

▼ Preliminaries

Let A be a finite set. We define the following notation:

a) $|A|$ is the number of elements of the set A.

b) $\mathcal{P}(A)$ is the set of all subsets of A, i.e.

$$X \in \mathcal{P}(A) \Leftrightarrow X \subseteq A$$

c) $\mathcal{P}_a(A)$ is the set of all subsets of A with a elements, i.e.

$$X \in \mathcal{P}_a(A) \Leftrightarrow (X \subseteq A \wedge |X| = a)$$

EXAMPLE

For $A = \{a, b, c\}$, we have:

$|A| = |\{a, b, c\}| = 3$

$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{c, a\}, \{a, b, c\}\}$

$\mathcal{P}_0(A) = \{\emptyset\}$

$\mathcal{P}_1(A) = \{\{a\}, \{b\}, \{c\}\}$

$\mathcal{P}_2(A) = \{\{a, b\}, \{b, c\}, \{c, a\}\}$

$\mathcal{P}_3(A) = \{\{a, b, c\}\}$

Def : Let $f : A \to B$ be a mapping from a set A to a set B. We say that

f one-to-one $\Leftrightarrow \forall x_1, x_2 \in A : (f(x_1) = f(x_2) \Rightarrow x_1 = x_2)$

interpretation: A mapping $f: A \to B$ is one-to-one if and only if each element of A is mapped to a distinct element of B.

▼ Graphs – Basic definitions

Def: A graph G is a triplet $G = (V(G), E(G), \psi_G)$ that consists of

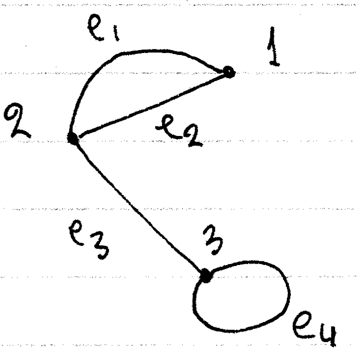
a) a set of vertices $V(G)$

b) a set of edges $E(G)$

c) an incidence mapping $\psi_G : E(G) \to P_1(V(G)) \cup P_2(V(G))$ that maps every edge to one or two vertices.

EXAMPLE

G:



$V(G) = \{1,2,3\}$
$E(G) = \{e_1, e_2, e_3, e_4\}$
$\psi_G(e_1) = \{1,2\}$
$\psi_G(e_2) = \{1,2\}$
$\psi_G(e_3) = \{2,3\}$
$\psi_G(e_4) = \{3\}$ ← a loop

Equivalently, ψ_G can be rewritten as:

$\psi_G = \{(e_1, \{1,2\}), (e_2, \{1,2\}), (e_3, \{2,3\}), (e_4, \{3\})\}$

Elementary definitions about graphs

Def : Let G be a graph. We say that

a) The vertex $u \in V(G)$ is *incident to* the edge $e \in E(G)$ if and only if the edge e connects u with itself or with another vertex:

$\forall u \in V(G) : \forall e \in E(G) : u \text{ incident to } e \Leftrightarrow u \in \psi_G(e)$

b) The edge $e \in E(G)$ is a *loop* if and only if it connects a vertex with itself.

$\forall e \in E(G) : (e \text{ loop} \Leftrightarrow |\psi_G(e)| = 1)$

Def : Let G be a graph and let $u \in V(G)$ be a vertex of G. The *degree* $d(u)$ of u is given by

$$d(u) = |\{e \in E(G) \mid u \in \psi_G(e)\}| + |\{e \in E(G) \mid u \in \psi_G(e) \wedge e \text{ loop}\}|$$

↳ The degree $d(u)$ is the number of edges $e \in E(G)$ to which u is incident, with the loops being counted twice. In the previous example:

$d(1) = 2 \wedge d(2) = 3 \wedge d(3) = 3$

Def : Let G be a graph. We define:

a) The *minimum degree* $\delta(G) = \min\limits_{u \in V(G)} d(u)$

b) The *maximum degree* $\Delta(G) = \max\limits_{u \in V(G)} d(u)$

Remark : It follows from this definition that

$$\forall u \in V(G) : \delta(G) \leqslant d(u) \leqslant \Delta(G)$$

Lemma : (**The Handshaking lemma**)

Let G be a graph. Then:

$$\sum_{u \in V(G)} d(u) = 2|E(G)|$$

↳ Although this is not a formal proof, a simple explanation of the handshaking lemma is that every edge is usually shared by two vertices. As a result, adding up the degrees of all vertices counts the edges twice. Although loops attach to only one vertex, the loop, by definition, adds 2 to the degree of that vertex, so loops are also counted twice

EXAMPLES

a) Show that it is not possible to define a graph with 3 vertices of degree 2 and 5 vertices of degree 3.

Solution

Assume a graph G exists with $V(G)=\{u_1,u_2,u_3,V_1,V_2,V_3,V_4,V_5\}$ such that

$$\begin{cases} \forall a\in[3]:\ d(u_a)=2 \\ \forall a\in[5]:\ d(v_a)=3 \end{cases}$$

It follows that

$$2|E(G)| = \sum_{u\in V(G)} d(u) = \sum_{a\in[3]} d(u_a) + \sum_{b\in[5]} d(v_a) =$$

$$= 3\cdot 2 + 5\cdot 3 = 6+15 = 21 \Rightarrow$$

$\Rightarrow |E(G)| = 21/2 \leftarrow$ contradiction since $|E(G)|$ is a natural number. It follows that G cannot be constructed.

b) A graph has twice as many vertices as edges. Show that there is at least one vertex with degree less than 2.

Solution

Let G be the graph with $2|E(G)| = |V(G)|$. Then:

$$2|E(G)| = \sum_{u\in V(G)} d(u) \geqslant \sum_{u\in V(G)} \delta(G) = \delta(G)\,|V(G)| =$$

$$= \delta(G)\, 2|E(G)| \Rightarrow 2|E(G)| \geqslant \delta(G)\, 2|E(G)| \Rightarrow$$

$$\Rightarrow \delta(G) \leqslant 1 \Rightarrow \delta(G) = 0 \vee \delta(G) = 1 \Rightarrow$$

$\Rightarrow \exists u \in V(G): (d(u)=0 \vee d(u)=1)$

$\Rightarrow \exists u \in V(G): d(u) \not\geq 2.$

c) Let G be a graph with $\Delta(G)=2$. Show that the graph cannot have more edges than vertices.

Solution

Since $\Delta(G)=2 \Rightarrow \forall u \in V(G): d(u) \leq 2$

$$\Rightarrow 2|E(G)| = \sum_{u\in V(G)} d(u) \leq \sum_{u \in V(G)} \Delta(G) = \Delta(G)|V(G)|$$

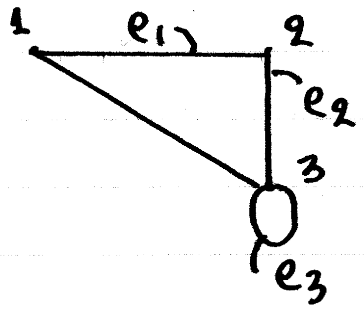
$$= 2|V(G)| \Rightarrow 2|E(G)| \leq 2|V(G)| \Rightarrow$$

$\Rightarrow |E(G)| \leq |V(G)|.$

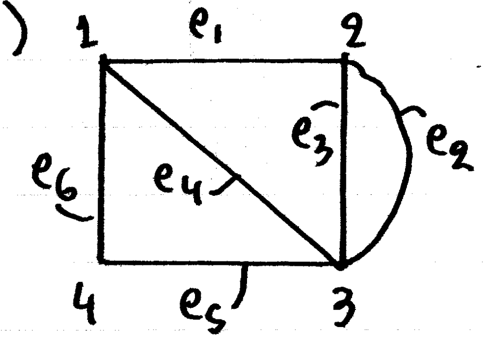
EXERCISES

① For the following graphs, list $V(G)$, $E(G)$, and the values of the incidence mapping ψ_G:

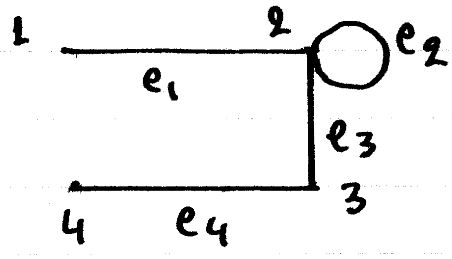
a)



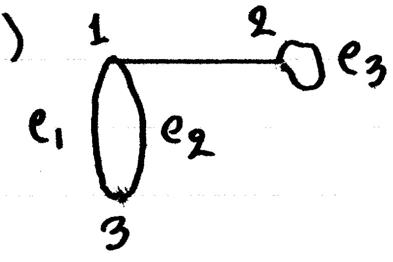
b)



c)



d)



② For the graphs of the previous exercise, list the degrees of each vertex and write $\delta(G)$ and $\Delta(G)$.

③ Show that it is not possible to create a graph with 9 vertices such that the degree of every vertex is 3.

(4) Show that it is not possible to create a graph with 7 vertices of degree 3 and 2 vertices of degree 2.

(5) Let G be a graph with 10 vertices such that

$$\delta(G) = \Delta(G) = 2$$

How many edges does G have?

(6) Let G be a graph with $|V(G)| = 8$ such that $\Delta(G) = 4$. Show that $|E(G)| < 36$.

(7) Let G be a graph such that $|V(G)| = |E(G)|$. Show that $\delta(G) < 3$

(8) Let G be a graph with $\Delta(G) = 4$. Show that

$$|E(G)| \leqslant 2|V(G)|$$

(9) A graph with 4 edges has a vertex with degree 4, a vertex with degree 1 and one more vertex. What is the degree of the third vertex?

▼ Types of graphs

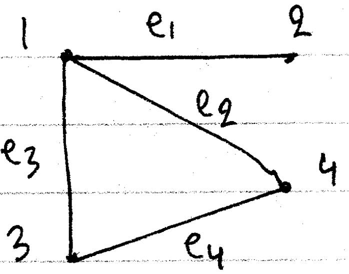
① → Simple graphs

<u>Def</u> : A graph G is <u>simple</u> if and only if it has no loops and no multiple edges, i.e.

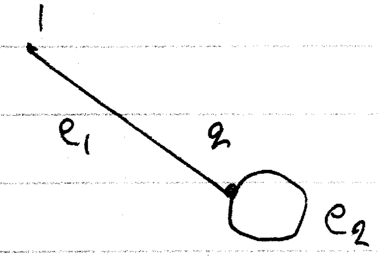
$$G \text{ simple} \Leftrightarrow \begin{cases} \forall e \in E(G) : |\psi_G(e)| = 2 \\ \psi_G \text{ is one-to-one} \end{cases}$$

EXAMPLE

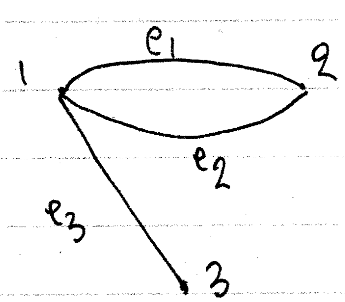
G_1:



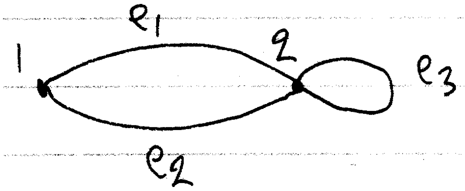
G_2:



G_3:



G_4:



G_1 is simple.

G_2, G_3, G_4 are NOT simple.

② → Regular graphs

Def: Let G be a graph. We say that

a) G is regular if and only if all vertices have the same degree, i.e.

G regular $\Leftrightarrow \forall u_1, u_2 \in V(G): d(u_1) = d(u_2)$

$\Leftrightarrow \exists a \in \mathbb{N}: \forall u \in V(G): d(u) = a$

b) G is a-regular if and only if all vertices have degree equal to a, i.e

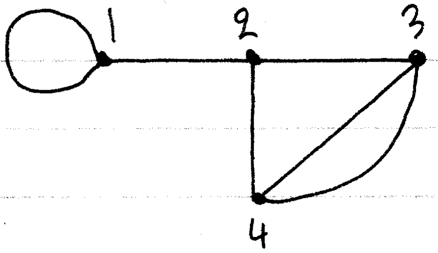
G a-regular $\Leftrightarrow \forall u \in V(G): d(u) = a$

Remark: Note the negation of this definition:

G not regular $\Leftrightarrow \exists u_1, u_2 \in V(G): d(u_1) \neq d(u_2)$

EXAMPLE

G:



$d(1) = d(2) = d(3) = d(4) = 3 \Rightarrow$

$\Rightarrow \forall u \in V(G): d(u) = 3$

$\Rightarrow G$ 3-regular

Also: G is regular.

3 → Complete graphs

A complete graph is a simple graph such that any two distinct vertices are connected by an edge. The formal definition reads:

Def: Let G be a graph. We say that

$$G \text{ complete} \Longleftrightarrow \begin{cases} G \text{ simple} \\ \forall u_1, u_2 \in V(G) : (u_1 \neq u_2 \Rightarrow \exists e \in E(G) : \psi_G(e) = \{u_1, u_2\}) \end{cases}$$

Given the number of vertices n, there is a unique graph such that

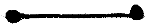
$$\begin{cases} G \text{ complete} \\ |V(G)| = n \end{cases}$$

which we denote as K_n.

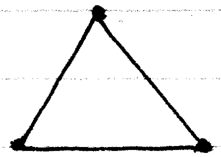
EXAMPLES

K_1:

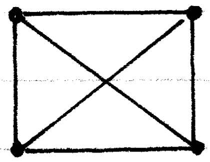
K_2:



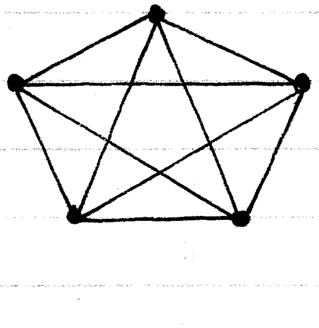
K_3:



K_4:



K_5:



④ → Null graphs

A null graph is a graph with no edges. The formal definition is:

Def: Let G be a graph. We say that G is a null graph $\Longleftrightarrow E(G) = \emptyset$

The null graph with n vertices is unique and denoted as N_n.

⑤ → The path graph P_n

Def: We distinguish between the following cases.

Case 1: For $n = 1$, we define

$$V(P_1) = \{u_1\} \wedge E(P_1) = \emptyset \wedge \psi_{P_1} = \emptyset$$

Case 2: For $n \geqslant 2$, we define

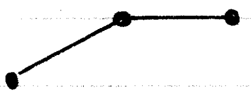
$$\begin{cases} V(P_n) = \{u_1, u_2, \dots, u_n\} \\ E(P_n) = \{e_1, e_2, \dots, e_{n-1}\} \\ \forall k \in [n-1] : \psi_{P_n}(e_k) = \{u_k, u_{k+1}\} \end{cases}$$

EXAMPLES

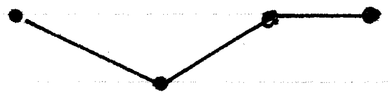
P_1:

P_2:

P_3:



P_4:



6 → The cycle graph C_n

Def : We distinguish between the following cases:

Case 1: For $n=1$, we have

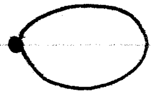
$$\begin{cases} V(C_1)=\{u_1\} \\ E(C_1)=\{e_1\} \\ \psi_{C_1}(e_1)=\{u_1\} \end{cases}$$

Case 2: For $n\geq 2$, we have

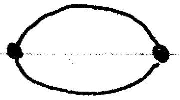
$$\begin{cases} V(C_n)=\{u_1,u_2,\dots,u_n\} \\ E(C_n)=\{e_1,e_2,\dots,e_n\} \\ \forall k\in[n]: \psi_{C_n}(e_k)=\{u_k,u_{k+1}\} \\ \psi_{C_n}(e_n)=\{u_n,u_1\} \end{cases}$$

EXAMPLES

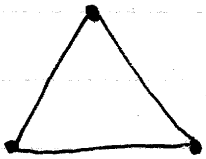
C_1 :



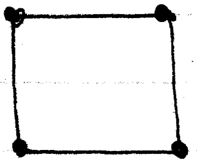
C_2 :



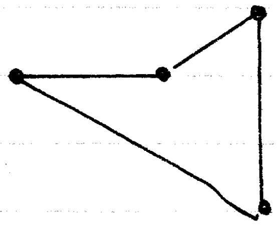
C_3 :



C_4 :



C_5 :



⑦ → Bipartite graphs

- A graph G is called *bipartite* if and only if its vertex $V(G)$ can be partitioned to two sets V_1 and V_2 such that every edge of G connects a vertex in V_1 with a vertex in V_2. The formal definition is:

Def: Let G be a graph. We say that

a) G *bipartite with vertex partition* $V_1, V_2 \iff$

$$\begin{cases} V_1 \cup V_2 = V(G) \\ V_1 \cap V_2 = \varnothing \\ \forall e \in E(G) : \begin{cases} |\psi_G(e) \cap V_1| = 1 \\ |\psi_G(e) \cap V_2| = 1 \end{cases} \end{cases}$$

b) G bipartite $\iff \exists V_1, V_2 \in \mathcal{P}(V(G)) :$ G bipartite with vertex partition V_1, V_2.

- A complete bipartite graph is bipartite with some vertex partition V_1, V_2, simple, and every vertex of V_1 is connected with every vertex of V_2 with exactly one edge. The formal definition is:

Def: Let G be a graph. We say that:

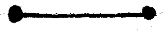
G *complete bipartite with vertex partition* $V_1, V_2 \iff$

$$\iff \begin{cases} G \text{ bipartite with vertex partition } V_1, V_2 \\ G \text{ simple} \\ \forall u_1 \in V_1 : \forall u_2 \in V_2 : \exists e \in E(G) : \psi_G(e) = \{u_1, u_2\} \end{cases}$$

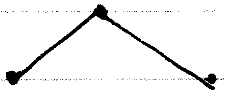
- There is a unique graph G such that
$$\begin{cases} G \text{ complete bipartite with vertex partition } V_1, V_2 \\ |V_1| = n \land |V_2| = m \end{cases}$$
with $n, m \in \mathbb{N}^*$, and it is denoted as $K_{n,m}$.

EXAMPLES

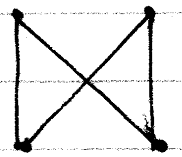
$K_{1,1}$:



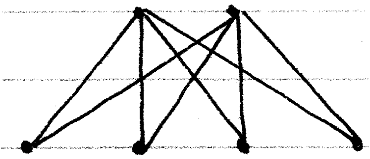
$K_{1,2}$:



$K_{2,2}$:



$K_{2,4}$:



EXAMPLES

a) Evaluate $\delta(K_{a,b})$, $\Delta(K_{a,b})$, and $|E(K_{a,b})|$ for the complete bipartite graph $K_{a,b}$.

Solution

Let $V(K_{a,b}) = V_1 \cup V_2$ with $|V_1| = a$ and $|V_2| = b$.

Each vertex of V_1 connects to all vertices of V_2, therefore

$\forall u \in V_1 : d(u) = |V_2| = b.$ (1)

Similarly, each vertex of V_2 connects to all vertices of V_1, and therefore:

$\forall u \in V_2 : d(u) = |V_1| = a$ (2)

It follows that

$$\delta(K_{a,b}) = \min_{u \in V(K_{a,b})} d(u) = \min\{a, b\}$$

$$\Delta(K_{a,b}) = \max_{u \in V(K_{a,b})} d(u) = \max\{a, b\}$$

$$2|E(K_{a,b})| = \sum_{u \in V(K_{a,b})} d(u) = \sum_{u \in V_1} d(u) + \sum_{u \in V_2} d(u) =$$

$$= |V_1| b + |V_2| a = ab + ba = 2ab \Rightarrow$$

$\Rightarrow |E(K_{a,b})| = ab.$

b) Show that P_{10} is not regular.

Solution

Let $V(P_{10}) = \{u_1, u_2\} \cup V_0$ with u_1, u_2 the endpoint vertices and $V_0 = \{v_1, v_2, \dots, v_8\}$ the interior vertices. We note that $d(u_1) = 1$ and $d(v_1) = 2$. It follows that

$$\exists u,v \in V(P_{10}) : d(u) \neq d(v)$$

$$\Rightarrow \overline{\forall u,v \in V(P_{10}) : d(u) = d(v)}$$

$\Rightarrow$ P_{10} is not regular.

c) Show that if G is regular, then

$$|E(G)| = \frac{\delta(G)\,|V(G)|}{2}$$

Solution

Assume G is regular. Then

$$G \text{ regular} \Rightarrow \forall u,v \in V(G) : d(u) = d(v)$$

$$\Rightarrow \exists a \in \mathbb{N} : \forall u \in V(G) : d(u) = a$$

It follows that

$$\delta(G) = \min_{u \in V(G)} d(u) = \min_{u \in V(G)} a = a$$

and therefore:

$$|E(G)| = \frac{1}{2} \sum_{u \in V(G)} d(u) = \frac{1}{2} \sum_{u \in V(G)} a =$$

$$= \frac{a\,|V(G)|}{2} = \frac{\delta(G)\,|V(G)|}{2}$$

EXERCISES

10 Draw the following graphs:

a) K_4
b) K_5
c) K_6
d) $K_{1,3}$
e) $K_{2,2}$
f) $K_{3,3}$
g) P_4
h) C_3
i) C_4

11 Which of the graphs in the previous exercise are regular?

12 For a,b integers $a>0$ and $b>0$ evaluate the following:

a) $\delta(K_a)$
b) $\delta(K_{a,b})$
c) $\delta(P_a)$
d) $\delta(C_a)$
e) $\Delta(K_a)$
f) $\Delta(K_{a,b})$
g) $\Delta(P_a)$
h) $\Delta(C_a)$
i) $|E(K_a)|$
j) $|E(K_{a,b})|$
k) $|E(P_a)|$
l) $|E(C_a)|$

[You can check your general answers by testing them when $a=2$, $b=3$ or $a=4$, $b=3$]

13 Show that

$K_{a,b}$ regular $\Leftrightarrow a=b$

(14) Show that $K_{5,7}$ is not regular.

(15) Show that
G regular $\Leftrightarrow \delta(G) = \Delta(G)$.

(16) Let G be a bipartite graph with bipartition $V(G) = V_1 \cup V_2$.
If $|V_1| = a$ and $|V_2| = a+2$
show that

$$|E(G)| \leq a^2 + 2a$$

(17) Show that we cannot build a bipartite graph with bipartition $V(G) = V_1 \cup V_2$ such that $|V_1| = 4$ and $|V_2| = 3$ and $|E(G)| > 14$.

▼ Graph operations

We define the following graph operations.

① → Induced subgraph

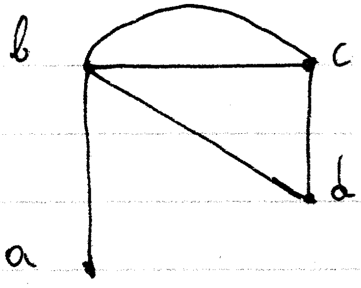
Def : Let G be a graph and let $V_0 \subseteq V(G)$. We define the vertex induced subgraph $G[V_0]$ such that

$$V(G[V_0]) = V_0$$
$$E(G[V_0]) = \{e \in E(G) \mid \psi_G(e) \subseteq V_0\}$$
$$\forall e \in E(G[V_0]) : \psi_{G[V_0]}(e) = \psi_G(e)$$

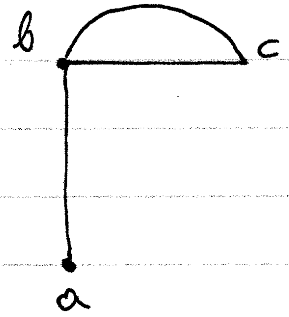
- Intuitively, the vertex induced subgraph $G[V_0]$ consists of the vertices V_0, the edges that are incident only to vertices in V_0, connected similarly as in G.

EXAMPLE

G:



$G[\{a,b,c\}]$:



Note that the removal of the vertex d removes the two edges that are incident to it.

② → Vertex subtraction

Def : Let G be a graph and let $V_0 \subseteq V(G)$. We define the graph $G - V_0$ such that

$$G - V_0 = G[V(G) - V_0]$$

Intuitively, $G - V_0$ is the graph obtained by deleting from G, the vertices in V_0 and all edges to which these vertices are incident.

③ → Edge-induced subgraph

Def : Let G be a graph and let $E_0 \subseteq E(G)$. We define the graph $G[E_0]$ such that

$$\begin{cases} V(G[E_0]) = V(G) \\ E(G[E_0]) = E_0 \\ \forall e \in E_0 : \psi_{G[E_0]}(e) = \psi_G(c) \end{cases}$$

Intuitively, $G[E_0]$ consists of all the vertices of the original graph G but only the edges that belong to E_0.

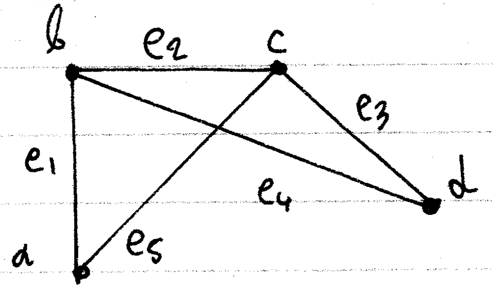
④ → Edge subtraction

Def : Let G be a graph and let $E_0 \subseteq E(G)$. We define the graph $G - E_0$ as:
$G - E_0 = G[E(G) - E_0]$

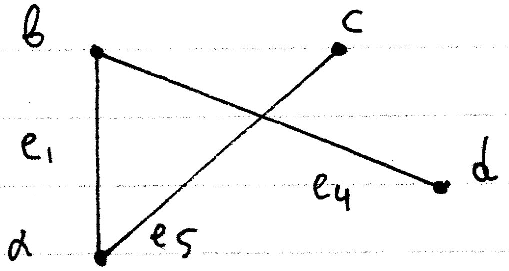
- Note that, unlike vertex subtraction, subtracting edges does not remove vertices under any circumstances.

EXAMPLE

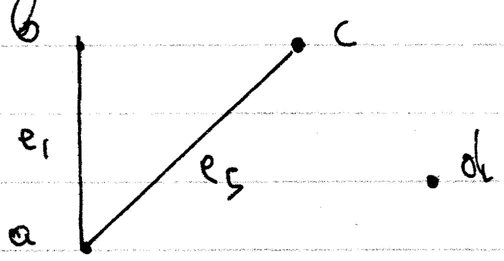
G:



$G - \{e_2, e_3\}$:



$G - \{e_2, e_3, e_4\}$:



⑤ → Graph union

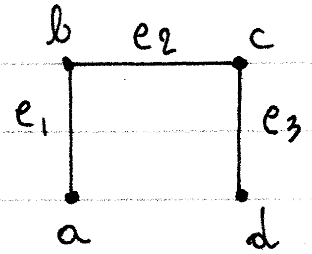
A necessary condition for defining the graph union $G_1 \cup G_2$ of two graphs G_1, G_2 is that G_1, G_2 should not share any edges, though they may share vertices. The formal definition is:

Def : Let G_1, G_2 be two graphs such that $E(G_1) \cap E(G_2) = \emptyset$. We define the graph union $G = G_1 \cup G_2$ such that

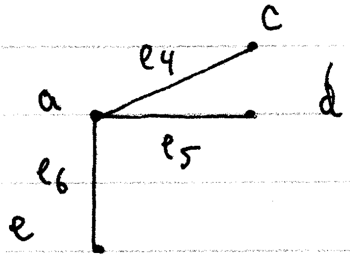
$$\begin{cases} V(G) = V(G_1) \cup V(G_2) \\ E(G) = E(G_1) \cup E(G_2) \\ \forall e \in E(G) : \psi_G(e) = \begin{cases} \psi_{G_1}(e) & , \text{if } e \in E(G_1) \\ \psi_{G_2}(e) & , \text{if } e \in E(G_2) \end{cases} \end{cases}$$

EXAMPLE

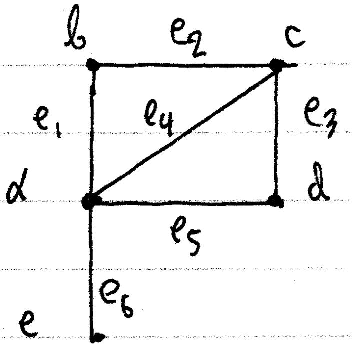
G_1 :



G_2 :



$G_1 \cup G_2$:



This definition generalizes to the union of n graphs as follows:

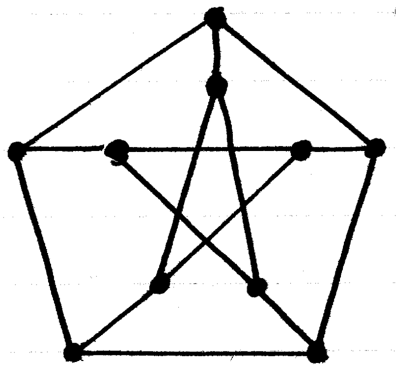
Def : Let $G_1, G_2, \ldots, G_n$ be graphs such that
$\forall k, m \in [n] : (k \neq m \Rightarrow E(G_k) \cap E(G_m) = \emptyset)$

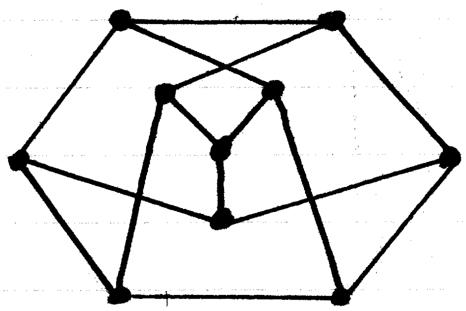
We define the graph $G = G_1 \cup G_2 \cup \cdots \cup G_n$ such that:

$$\left\{ \begin{aligned} & V(G) = \bigcup_{a \in [n]} V(G_a) = V(G_1) \cup V(G_2) \cup \cdots \cup V(G_n) \\ & E(G) = \bigcup_{a \in [n]} E(G_a) = E(G_1) \cup E(G_2) \cup \cdots \cup E(G_n) \\ & \forall e \in E(G) : \forall k \in [n] : (e \in E(G_k) \Rightarrow \psi_G(e) = \psi_{G_k}(e) \end{aligned} \right.$$

EXERCISES

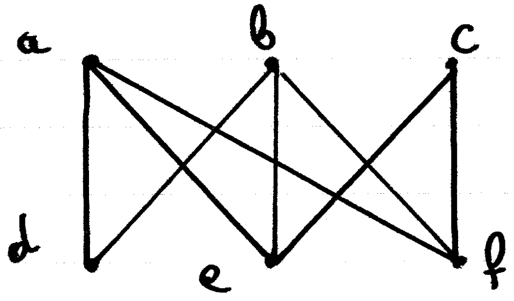
(18) Show that the following graphs are isomorphic





(Hint: Look at the "cycles")

(19) Consider the graph $K_{3,3} = G$



Draw the following:

a) $G[\{a,b,d\}]$

b) $G[\{a,d,e,f\}]$

c) $G[\{a,b,d,e\}]$

d) $G - \{a\}$

e) $G - \{a,b\}$

f) $G - \{a,d\}$

g) $G - \{c,d,e\}$

h) $G - \{d,e,f\}$

i) $G - \{a,c,e,f\}$

20 In the previous exercise, let

$G_1 = G - \{a, c, e, f\}$

$G_2 = G[\{a, b, e\}]$

Draw $G_1 \cup G_2$.

[Hint : List $V(G_1)$, $E(G_1)$, $V(G_2)$, $E(G_2)$ first].

21 In the previous exercise show that

$G[\{a, d\}] \cup G[\{b, e\}] \neq G[\{a, d, b, e\}]$

▼ Connected graphs

↳ Walks, trails, paths

- Let G be a graph. A walk w is a sequence of alternating vertices and edges of the form
$$w = (v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n)$$
such that
$$\forall k \in [n] : \psi_G(e_k) = \{v_{k-1}, v_k\}.$$

- Features of a walk.

a) Starting point : $s(w) = v_0$

b) Terminal point : $t(w) = v_n$

c) $v_k(w) = v_k$

$e_k(w) = e_k$

d) Vertex set : $V(w) = \{v_0, v_1, \dots, v_n\}$

e) Edge set : $E(w) = \{e_1, e_2, \dots, e_n\}$

f) Length : $\ell(w) = |E(w)| = n$.

- The set of all walks in G is denoted $W(G)$.

- A trail is a walk in which all the edges are different. A path is a walk in which all the edges and vertices are different.

▶ Thus, for $w \in W(G)$

a) w trail $\Leftrightarrow$

$\Leftrightarrow \forall m,n \in [\ell(w)] : (m \neq n \Rightarrow e_m(w) \neq e_n(w))$

b) w path $\Leftrightarrow$

$$\Leftrightarrow \begin{cases} w \text{ trail} \\ \forall m,n \in [\ell(w)] \cup \{0\} : (m \neq n \Rightarrow v_m(w) \neq v_n(w)) \end{cases}$$

- We define

  $T(G) = \{w \in W(G) \mid w \text{ is a trail}\}$

  $P(G) = \{w \in W(G) \mid w \text{ is a path}\}$

- Let $u,v \in V(G)$ be two vertices of G with $u \neq v$. Then we define

  a) Set of all trails that connect u to v

  $T(G, u \to v) = \{w \in T(G) \mid s(w) = u \wedge t(w) = v\}$

  b) Set of all paths that connect u to v

  $P(G, u \to v) = \{w \in P(G) \mid s(w) = u \wedge t(w) = v\}$.

- Note that $W(G)$ is an infinite set (i.e. you can go back and forth between two vertices indefinitely) but $T(G)$ and $P(G)$ are both finite sets. (i.e you will run out of combinations of distinct edges and/or vertices).

↳ Connected graphs

- A graph G is connected if for any two not-equal vertices $u, v \in V(G)$, there is at least one path from u to v.

$$\boxed{G \text{ connected} \Leftrightarrow \forall\, u,v \in V(G) : (u \neq v \Rightarrow |P(G, u \to v)| \geqslant 1)}$$

- The following graphs are connected:
  a) Complete graph K_n
  b) Path graph P_n
  c) Cycle graph C_n
  d) The bipartite graph $K_{m,n}$.

↳ Graph components

<u>Thm</u> : Let G be a graph which is not connected. Then the vertex set $V(G)$ can pbe partitioned to w pieces $V_1, V_2, \ldots, V_w$ such that

a) $\forall\, m,n \in [w] : m \neq n \Rightarrow V_m \cap V_n = \emptyset$

b) $V_1 \cup V_2 \cup \cdots \cup V_w = V(G)$

c) $G[V_n]$ connected , $\forall n \in [\omega]$
d) $G[V_1] \cup G[V_2] \cup \cdots \cup G[V_\omega] = G$

↳ The subgraphs $G[V_1], \dots, G[V_\omega]$ are called <u>components</u> of G.

- $\omega(G) =$ the number of components of G.
- Obviously:
  G connected $\Leftrightarrow$ $\omega(G) = 1$
  G not connected $\Leftrightarrow$ $\omega(G) > 1$.

↱ <u>Bridges</u> .

<u>Thm</u> : For any graph G :

$$\forall e \in E(G): \ \omega(G) \leqslant \omega(G - \{e\}) \leqslant \omega(G) + 1$$

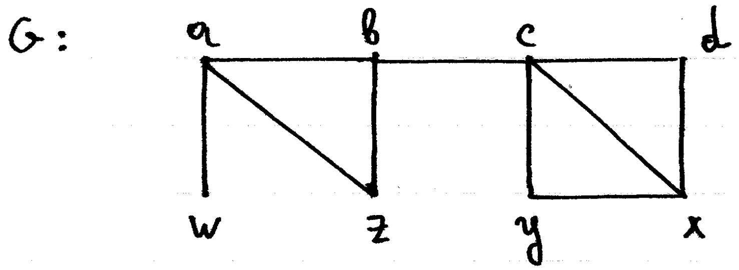
i.e. removing an edge may or may not increase the number of components by 1.

<u>Remark</u> : This theorem cannot be generalized to the deletion of vertices.

- Let G be a graph. An edge $e \in E(G)$ is called a <u>bridge</u> if the deletion of e increases the number of components in the resulting graph.

$$\boxed{e \in E(G) \text{ bridge} \Leftrightarrow \omega(G - \{e\}) > \omega(G)}$$

<u>example</u>



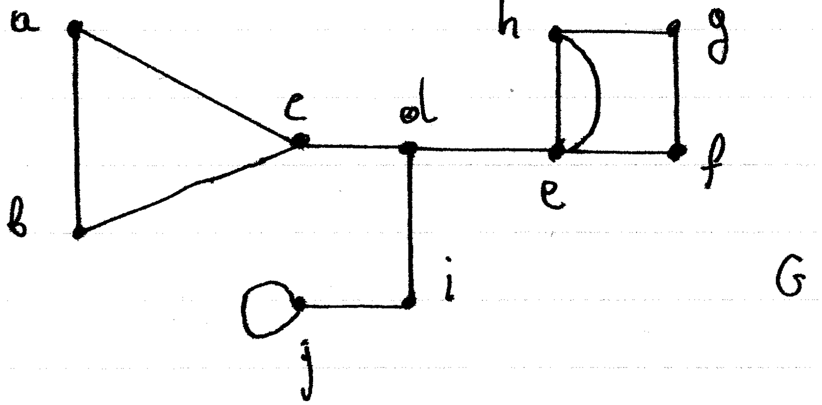
The edges aw and bc are bridges.

- Let G be a connected graph. We say that
  a) G is <u>weakly-linked</u> if it has at least one bridge
  b) G is <u>strongly-linked</u> if it has no bridges.

- Thus:
  a) G strongly-linked $\Leftrightarrow \forall e \in E(G)$: $G - \{e\}$ connected
  b) G weakly-linked $\Leftrightarrow \begin{cases} G \text{ connected} \\ \exists e \in E(G): G - \{e\} \text{ not connected.} \end{cases}$

EXERCISES

(22) Consider the following graph



a) List the components of the following graphs:

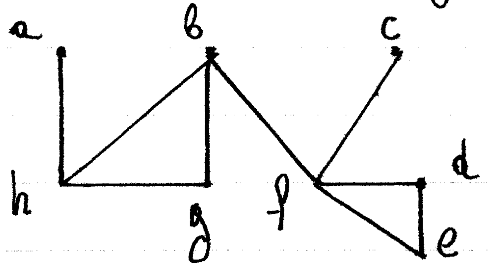
$G_1 = G - \{c\}$ $\quad$ $G_4 = G - \{e\}$

$G_2 = G - \{d\}$ $\quad$ $G_5 = G - \{he, gf\}$

$G_3 = G - \{i\}$ $\quad$ $G_6 = G - \{di\}$

b) What are the bridges of the graph G?

(23) Consider the following graph



a) List the components of the following graphs:

$G_1 = G - \{bf\}$ $\quad$ $G_4 = G - \{bh, bg\}$

$G_2 = G - \{g\}$ $\quad$ $G_5 = G - \{h\}$

$G_3 = G - \{b, g\}$ $\quad$ $G_6 = G - \{f\}$

b) What are the bridges of the graph G?

(24) Let G be a connected graph and let $e \in E(G)$. Show that

$$\omega(G - \{e\}) \leqslant 2$$

▼ The Laplacian matrix

- Let G be a graph with $n = |V(G)|$ vertices:

  $V(G) = \{v_1, v_2, \ldots, v_n\}$

  The Laplacian matrix L_G is defined as

$$\boxed{(L_G)_{ab} = \begin{cases} d(v_a) & \text{, if } a = b \\ -1 & \text{, if } a \neq b \text{ and } v_a \longleftrightarrow v_b \\ 0 & \text{, otherwise.} \end{cases}}$$

- If $\omega(G)$ is the number of components of G then the characteristic polynomial of L_G has a common factor $\lambda^{\omega(G)}$

  (i.e. 0 is a root with multiplicity $\omega(G)$)

  Thus

$$\boxed{\begin{array}{l} \det(L_G - \lambda I) = \lambda^{\omega(G)} f(\lambda) \\ \text{with } f(0) \neq 0 \end{array}}$$

▼ Graph connectivity

Edge connectivity $\lambda(G)$

Def: Let G be a graph and let $E_0 \subseteq E(G)$. We say that

E_0 is an edge cutset of $G \Leftrightarrow$

$$\Leftrightarrow \begin{cases} G - E_0 \text{ not connected} \\ \forall E_1 \in \mathcal{P}(E_0) : (E_1 \neq E_0 \Rightarrow G - E_1 \text{ connected}) \end{cases}$$

- The smallest number of edges needed to construct a cutset E_0 of G is the edge-connectivity $\lambda(G)$ of G. More formally,

$$\boxed{\lambda(G) = \min\{|E_0| \mid E_0 \in \mathcal{P}(E(G)) \wedge E_0 \text{ edge cut-set of } G\}}$$

Vertex connectivity $\kappa(G)$

Def: Let G be a graph and let $V_0 \subseteq V(G)$. We say that V_0 is a vertex cutset of $G \Leftrightarrow$

$$\Leftrightarrow \begin{cases} G - V_0 \text{ not connected} \\ \forall V_1 \in \mathcal{P}(V_0) : (V_1 \neq V_0 \Rightarrow G - V_1 \text{ connected}) \end{cases}$$

- The smallest number of vertices needed to construct a vertex cutset V_0 of G is the vertex connectivity $\kappa(G)$ of G

$$\boxed{\kappa(G) = \min\{|V_0| \mid V_0 \in \mathcal{P}(V(G)) \land V_0 \text{ vertex cutset of } G\}}$$

↳ Note that

G not connected $\iff \lambda(G) = \kappa(G) = 0$

G weakly linked $\iff \lambda(G) = 1$

G strongly linked $\iff \lambda(G) > 1$

A property of connectivity

Recall that $\delta(G)$ is the minimum degree of G:

$$\delta(G) = \min\{d(u) \mid u \in V(G)\}$$

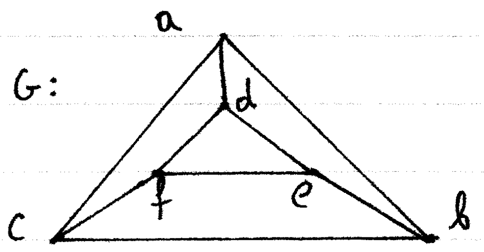
It can be shown that:

Thm: Let G be a graph. Then:

$$G \text{ connected} \implies \kappa(G) \leq \lambda(G) \leq \delta(G) \leq \frac{2|E(G)|}{|V(G)|}$$

EXAMPLE

Calculate the vertex connectivity $\kappa(G)$ and edge connectivity $\lambda(G)$ for the following graph G:



Solution

First we note that

$d(a) = d(b) = d(c) = d(d) = d(e) = d(f) = 3 \Rightarrow$

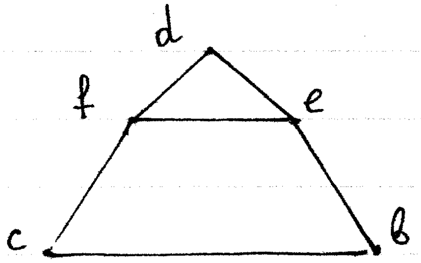
$\Rightarrow \delta(G) = \min\limits_{u \in V(G)} d(u) = 3 \Rightarrow \kappa(G) \leq \delta(G) = 3 \Rightarrow \kappa(G) \leq 3$

$\Rightarrow \kappa(G) = 0 \vee \kappa(G) = 1 \vee \kappa(G) = 2 \vee \kappa(G) = 3.$

Since G connected $\Rightarrow \kappa(G) > 0$.

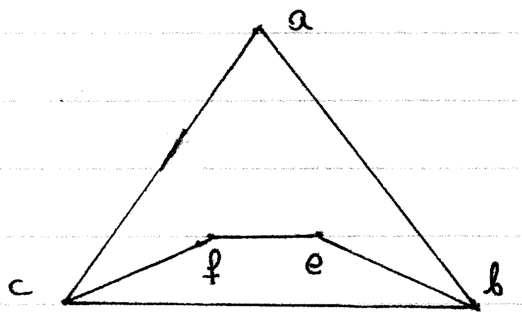
- Try deleting one vertex

a) For $G - \{a\}$ we have:



which is connected. $G - \{b\}$, $G - \{c\}$ are similarly connected.

b) For $G-\{d\}$ we have:

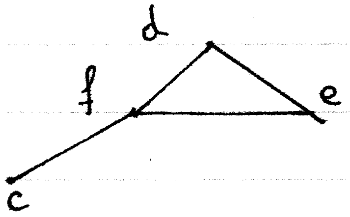


which is connected, and by symmetry, $G-\{e\}$ and $G-\{f\}$ are also connected.

It follows from (a) and (b) that $\kappa(G) > 1$.

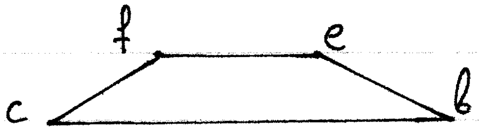
- Try deleting two vertices

a) For $G-\{a,b\}$ we have:



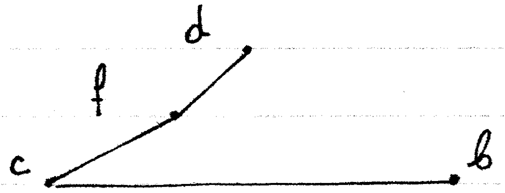
which is still connected. By symmetry, $G-\{b,c\}$ and $G-\{c,a\}$ are also connected.

b) For $G-\{a,d\}$ we have:



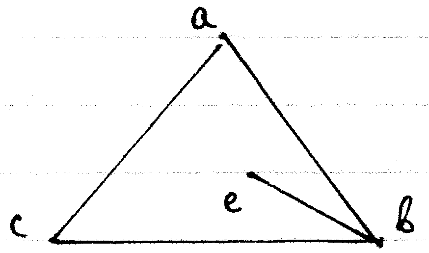
which is still connected. By symmetry, $G-\{b,e\}$ and $G-\{c,f\}$ are also connected

c) For $G-\{a,e\}$ we have:



which is still connected. By symmetry, $G-\{a,f\}$, $G-\{b,d\}$, $G-\{b,f\}$, $G-\{c,d\}$, $G-\{c,e\}$ are also connected.

d) For $G - \{d, f\}$ we have:



which is connected, and by symmetry $G - \{d, e\}$ and $G - \{e, f\}$ are also connected.

From (a), (b), (c), (d) it follows that $\kappa(G) > 2$.

Since $2 < \kappa(G) \leqslant \delta(G) = 3 \Rightarrow \underline{\kappa(G) = 3}$

and since $\kappa(G) \leqslant \lambda(G) < \delta(G) \Rightarrow 3 \leqslant \lambda(G) \leqslant 3$

$\Rightarrow \underline{\lambda(G) = 3}$

EXERCISES

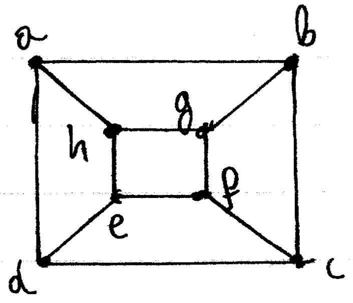
25 Consider the complete graph K_a
Let $u \in V(K_a)$. Show
a) Show that $K_a - \{u\} = K_{a-1}$
b) Show that
$\kappa(K_a) = \lambda(K_a) = \delta(K_a) = a-1$

26 Similarly, for the complete bipartite graph $K_{a,b}$ show that
$\kappa(K_{a,b}) = \lambda(K_{a,b}) = \delta(K_{a,b}) = \min\{a,b\}$

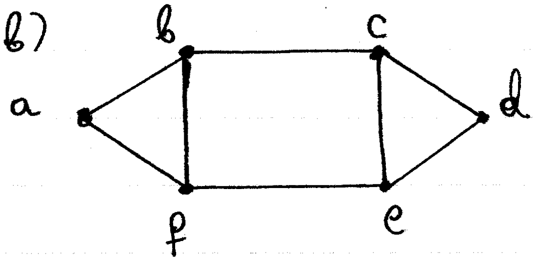
27 Show that
a) $\kappa(P_a) = \lambda(P_a) = \delta(P_a) = 1$
b) $\kappa(C_a) = \lambda(C_a) = \delta(C_a) = 2$

28 Calculate $\kappa(G)$ and $\lambda(G)$ for the following graphs:

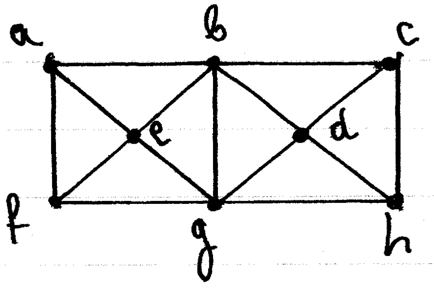
a)



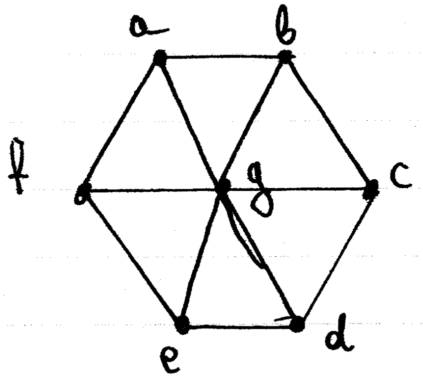
b)



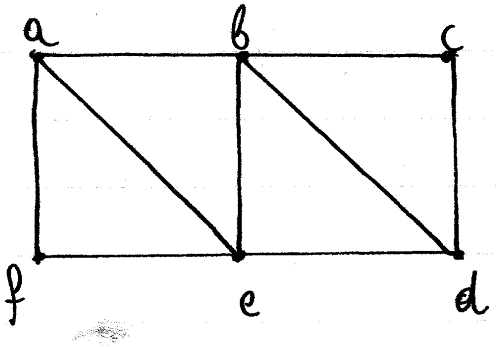
c)



d)



e)



V Eulerian graphs

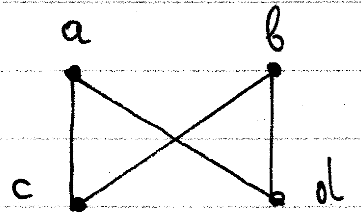
The Eulerian problem: Given a connected graph G, is there a walk that can visit every edge of the graph once and only once and return to the starting vertex at the end? If the answer is yes, we say that G is an Eulerian graph and the corresponding walk is an Eulerian trail.

Def: Let G be a connected graph. We say that

$$G \text{ Eulerian} \iff \exists w \in T(G) : E(w) = E(G) \wedge s(w) = t(w)$$

EXAMPLE

The graph $K_{2,2}$:



is Eulerian with Eulerian trail:

$w = (a, ac, c, cb, b, bd, d, da, a)$

Euler solved the Eulerian problem by introducing the definitions for graph, vertex degree, and proving the following theorem:

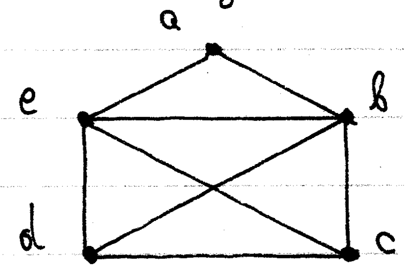
Thm: Let G be a connected graph. Then:

$$G \text{ Eulerian} \iff \forall u \in V(G) : \exists k \in \mathbb{N}^* : d(u) = 2k$$

EXAMPLE

Consider the graph

G :



$d(c) = |\{bc, cd, ce\}| = 3 \Rightarrow G$ not Eulerian.

EXAMPLE

A connected graph with 5 vertices and 4 edges has two vertices with degree 2. Show that the graph G is not Eulerian.

Solution

We assume that $|V(G)| = 5$ and $|E(G)| = 4$ with

$V(G) = \{u_1, u_2, u_3, u_4, u_5\}$

and $d(u_1) = d(u_2) = 2$. Define $a = d(u_3) \wedge b = d(u_4) \wedge c = d(u_5)$. From the handshaking lemma:

$$\sum_{u \in V(G)} d(u) = 2|E(G)| \Rightarrow d(u_1) + d(u_2) + d(u_3) + d(u_4) + d(u_5) = 2 \cdot 4$$

$$\Rightarrow 2 + 2 + \alpha + b + c = 8$$

$$\Rightarrow a + b + c = 4.$$

G connected $\Rightarrow \forall u \in V(G) : d(u) > 0$

$\Rightarrow a > 0 \wedge b > 0 \wedge c > 0.$

$\Rightarrow a \geqslant 1 \wedge b \geqslant 1 \wedge c \geqslant 1$

It follows that

$a+b+c = 4 \Leftrightarrow$

$\Leftrightarrow (a,b,c) \in \{(1,1,2),(1,2,1),(2,1,1)\}$

and therefore:

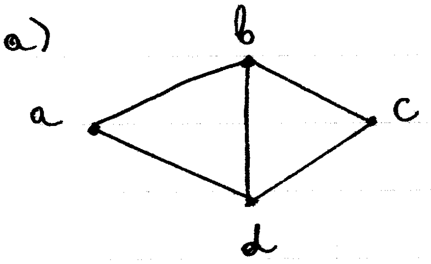
a odd $\vee$ b odd $\vee$ c odd $\Rightarrow$

$\Rightarrow$ G not Eulerian. $\square$

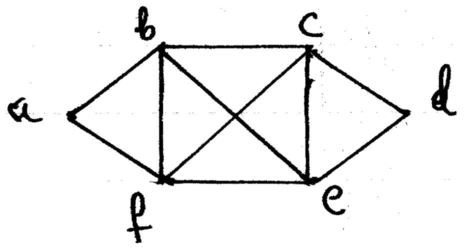
EXERCISES

(29) Which of the following graphs is Eulerian?

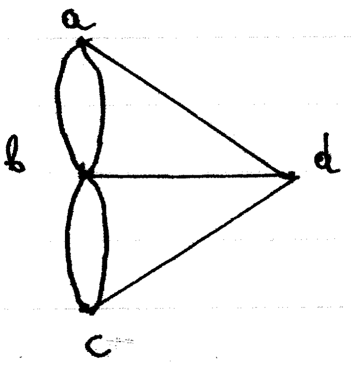
a)



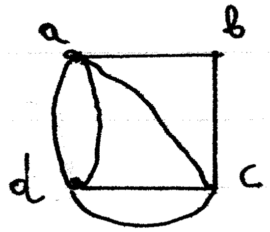
b)



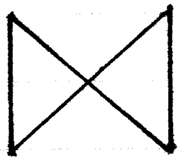
c)



d)

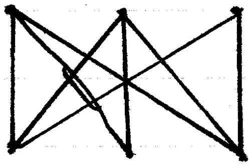


e)



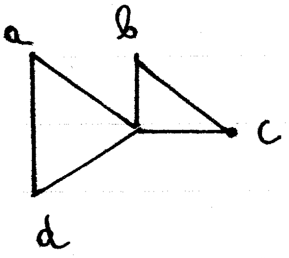
$K_{2,2}$

f)

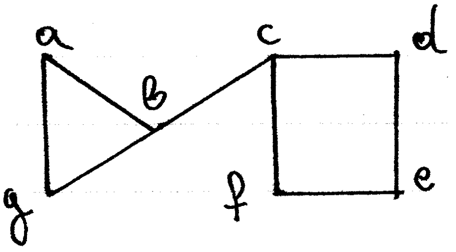


$K_{3,3}$

g)



h)



(30) Show that

a) K_a Eulerian $\Leftrightarrow$ a is odd

b) $K_{a,b}$ Eulerian $\Leftrightarrow$ a even $\wedge$ b even

c) $\forall a \in \mathbb{N}: (a \geq 2 \Rightarrow P_a$ not Eulerian$)$

d) $\forall a \in \mathbb{N}: (a \geq 3 \Rightarrow C_a$ Eulerian$)$

(31) A connected Eulerian graph has 3 vertices and 5 edges. Show that if one vertex has degree 4, then another vertex must have degree 2

(32) A connected graph with 4 edges and 4 vertices has 2 vertices of degree 2 Show that

a) G not Eulerian $\Rightarrow \exists u \in V(G): d(u) = 3$.

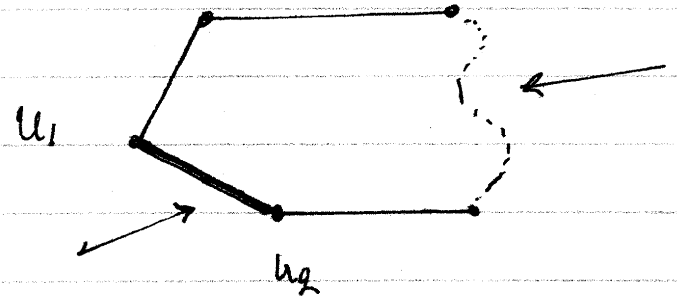
b) G Eulerian $\Rightarrow$ G regular.

(33) Show that a connected regular graph with an odd number of vertices is always Eulerian

(34) Show that a connected regular graph with odd number of edges and whose number of vertices is a multiple of 4 is never Eulerian.

Hamiltonian graphs

Hamilton's Problem : Let G be a connected graph. Can we construct a walk that visits every vertex of the graph *once and only once*, without using any edge more than once, and then close the walk with a direct edge from its terminal point back to its initial point? If yes, then we say that the graph is a *Hamiltonian graph*, the walk is a *Hamiltonian path*, and the walk together with the closing edge is a *Hamiltonian circuit.*



Hamiltonian path

closing edge

Recall that any walk where no edges or vertices are repeated is a path. The Hamiltonian circuit as a whole is not a path since the initial vertex is repeated once, as a terminal vertex. Thus, the reason for the distinction between the Hamiltonian path and the Hamiltonian circuit. Based on the above, we give the following definition:

Def : Let G be a connected graph. We say that G Hamiltonian $\Leftrightarrow$

$$\Leftrightarrow \exists u_1, u_2 \in V(G) : \exists w \in \mathcal{P}(G, u_1 \to u_2) : \begin{cases} u_1 \neq u_2 \wedge V(w) = V(G) \\ \exists e \in E(G) : \psi_G(e) = \{u_1, u_2\} \end{cases}$$

Here, w is the Hamiltonian path, u_1 the initial vertex, u_2 the terminal vertex and e the closing edge.

- Note that it is not necessary for the Hamiltonian circuit to visit all the edges.

● → Criteria for the Hamiltonian property

Noone has successfully solved the Hamiltonian problem by proving a practical necessary and sufficient condition. We have however the following partial results:

① → A necessary condition

Thm : Let G be a connected graph. Then:

$$G \text{ Hamiltonian} \Rightarrow \forall V_0 \in \mathcal{P}(V(G)) : (V_0 \neq V(G) \Rightarrow \omega(G - V_0) \leq |V_0|)$$

Intuitively, if the graph G is Hamiltonian, then if we subtract the vertices in $V_0 \subset V(G)$, then the resulting graph $G-V_0$ cannot have more components than the number of vertices in V_0.

- The <u>contrapositive</u> statement of this theorem can be used to show that a graph is not Hamiltonian

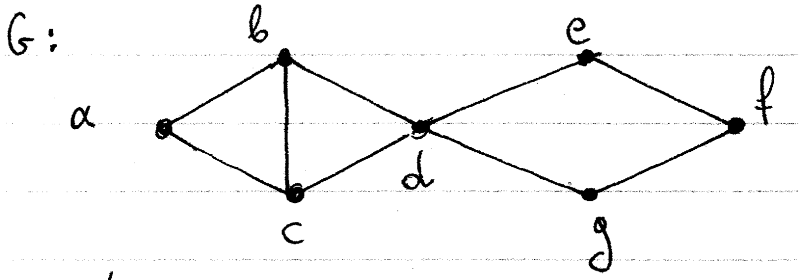
<u>Corollary</u>: Let G be a connected graph. Then
$(\exists V_0 \in \mathcal{P}(V(G)) : (V_0 \neq V(G) \wedge \omega(G-V_0) > |V_0|)) \Rightarrow G$ not Hamiltonian

↳ In general, proving a statement of the form $p \Rightarrow q$ also proves the contrapositive statement $\bar{q} \Rightarrow \bar{p}$. Negations can be calculated according to the following rules of Boolean logic:

Statement	It's negation
$\forall x \in A : p(x)$	$\exists x \in A : \overline{p(x)}$
$\exists x \in A : p(x)$	$\forall x \in A : \overline{p(x)}$
$p \wedge q$	$\bar{p} \vee \bar{q}$
$p \vee q$	$\bar{p} \wedge \bar{q}$
$p \Rightarrow q$	$p \wedge \bar{q}$
$p \Leftrightarrow q$	$p \veebar q$
$p \veebar q$	$p \Leftrightarrow q$

EXAMPLE

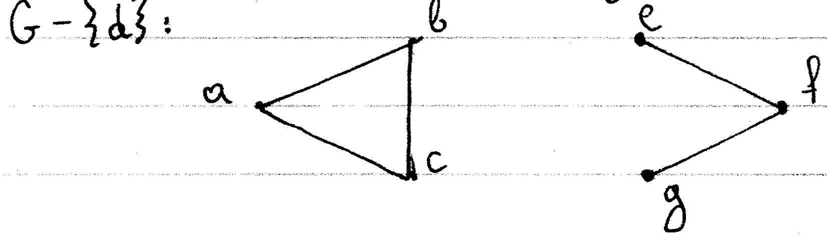
Show that the graph



is not Hamiltonian.

Solution

Subtracting the vertex d gives



It follows that

$G - \{d\} = G[\{a,b,c\}] \cup G[\{e,f,g\}] \Rightarrow$

$\Rightarrow w(G-\{d\}) = 2 > 1 = |\{d\}| \Rightarrow$

$\Rightarrow w(G-\{d\}) > |\{d\}| \Rightarrow$

$\Rightarrow$ G not Hamiltonian.

② → Ore's theorem

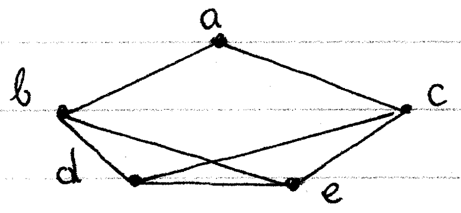
Thm: Let G be a graph. Then:

$$\left\{\begin{array}{l} G \text{ simple and connected} \\ |V(G)|\geq 3 \\ \forall u,v\in V(G) : (u,v \text{ not adjacent} \Rightarrow \\ \qquad\qquad \Rightarrow d(u)+d(v)\geq |V(G)|) \end{array}\right. \Rightarrow G \text{ is Hamiltonian}$$

EXAMPLE

Use Ore's theorem to show that

G :



is Hamiltonian

Solution

We note that

$$\left.\begin{array}{l} G \text{ is simple and connected.} \\ |V(G)| = |\{a,b,c,d,e\}| = 5 \geq 3 \\ d(a)+d(d) = 2+3 = 5 \geq |V(G)| \\ d(a)+d(e) = 2+3 = 5 \geq |V(G)| \\ d(b)+d(c) = 3+3 = 6 \geq |V(G) \end{array}\right\} \Rightarrow G \text{ is ....Hamiltonian}$$

③ → Dirac's theorem

Thm : Let G be a graph. Then

$$\left\{\begin{array}{l} G \text{ simple and connected} \\ |V(G)| \geq 3 \\ \delta(G) \geq (1/2)|V(G)| \end{array}\right. \Rightarrow G \text{ is Hamiltonian}$$

Proof

Assume that

$$\left\{\begin{array}{ll} G \text{ simple and connected} & (1) \\ |V(G)| \geq 3 & (2) \\ \delta(G) \geq (1/2)|V(G)| & (3) \end{array}\right.$$

Let $u,v \in V(G)$ be given and assume that u,v not adjacent. Then:

$$d(u)+d(v) \geq \delta(G)+\delta(G) = 2\delta(G) \geq 2[(1/2)|V(G)|] = |V(G)| \Rightarrow$$

$$\Rightarrow d(u)+d(v) \geq |V(G)|$$

It follows that

$$\forall u,v \in V(G) : (u,v \text{ not adjacent} \Rightarrow d(u)+d(v) \geq |V(G)|) \quad (4)$$

From Eq. (1), Eq. (2), Eq. (4), via Ore's theorem, it follows that G is Hamiltonian

④ → Bipartite graphs

Thm : Let G be a graph. Then

$$\left\{\begin{array}{l} G \text{ connected and bipartite} \\ \exists k \in \mathbb{N} : |V(G)| = 2k+1 \end{array}\right. \Rightarrow G \text{ not Hamiltonian}$$

Proof

Assume that

$$\begin{cases} G \text{ connected and bipartite} \\ \exists k \in \mathbb{N} : |V(G)| = 2k+1 \end{cases}$$

Since, G is bipartite, we choose $V_1 \subseteq V(G)$ and $V_2 \subseteq V(G)$ such that

$$\begin{cases} V_1 \cap V_2 = \emptyset \wedge V_1 \cup V_2 = V(G) \\ \forall e \in E(G) : \begin{cases} |\psi_G(e) \cap V_1| = 1 \\ |\psi_G(e) \cap V_2| = 1 \end{cases} \end{cases}$$

To show a contradiction, assume that G is Hamiltonian. Then, a Hamiltonian circuit must alternate between vertices in V_1 and vertices in V_2. Because each vertex can only be visited once, it follows that

$$|V_1| = |V_2| \Rightarrow$$
$$\Rightarrow |V(G)| = |V_1| + |V_2| = |V_1| + |V_1| = 2|V_1| \Rightarrow$$
$$\Rightarrow |V(G)| \text{ is even}$$
$$\Rightarrow |V(G)| \text{ not odd}$$

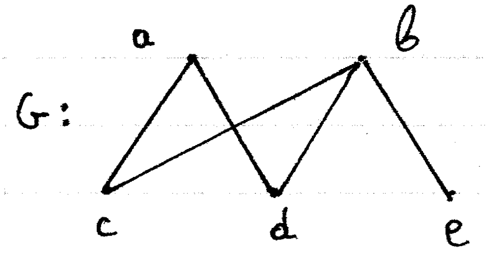
which contradicts the assumption

$$\exists k \in \mathbb{N} : |V(G)| = 2k+1$$

It follows that G is not Hamiltonian.

EXAMPLES

Show that the following graph is not Hamiltonian:



Solution

Note that for $V_1 = \{a, b\}$ and $V_2 = \{c, d, e\}$:

$$\forall e \in E(G): \begin{cases} |\psi_G(e) \cap V_1| = 1 \\ |\psi_G(e) \cap V_2| = 1 \end{cases} \Rightarrow G \text{ bipartite} \quad (1)$$

Furthermore:

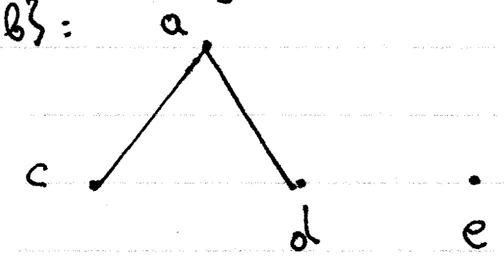
$$|V(G)| = |\{a, b, c, d, e\}| = 5 \Rightarrow |V(G)| \text{ odd} \quad (2)$$

From (1) and (2): G is not Hamiltonian.

2nd method

Consider the graph

$G - \{b\}$:



Since $G - \{b\} = G[\{a, c, d\}] \cup G[\{e\}] \Rightarrow$

$\Rightarrow \omega(G - \{b\}) = 2 > 1 = |\{b\}| \Rightarrow$

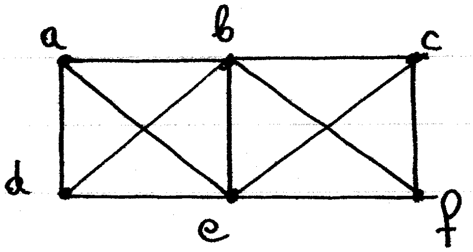
$\Rightarrow \omega(G - \{b\}) > |\{b\}|$

$\rightarrow$ G not Hamiltonian.

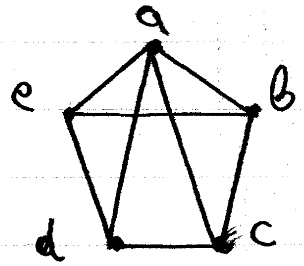
EXERCISES

35 Show that the following graphs are Hamiltonian

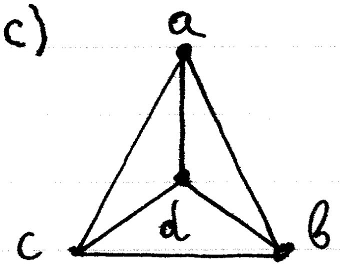
a)



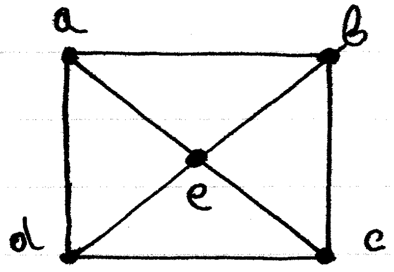
b)



c)

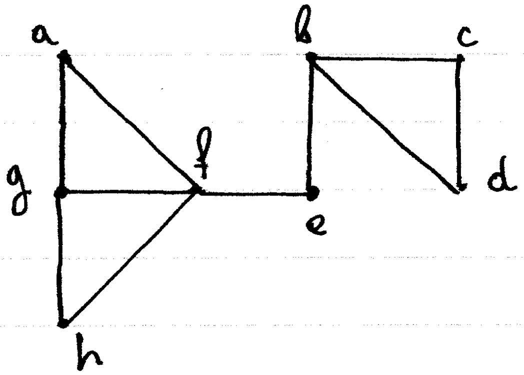


d)

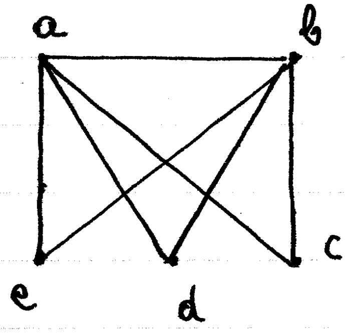


36 Show that the following graphs are not Hamiltonian

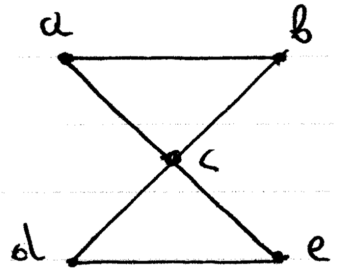
a)



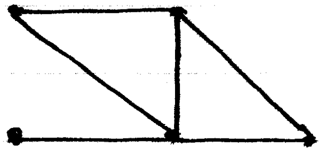
b)



c)



d)



(37) Show that K_a is Hamiltonian for all $a \geqslant 3$.

(38) Show that

a) $a = b \Rightarrow K_{a,b}$ Hamiltonian

b) $a \neq b \Rightarrow K_{a,b}$ not Hamiltonian

↳ It follows from this exercise that $K_{a,b}$ Hamiltonian $\Leftrightarrow a = b$.

(39) Let G be a graph with less than 7 vertices and vertex connectivity $\kappa(G) = 4$. Show that G is Hamiltonian.

(40) Show that a graph G with vertex connectivity $\kappa(G) = 1$ is not Hamiltonian

(41) Show that a strongly-linked graph with 4 vertices is always Hamiltonian.

EIGENVALUES AND EIGENVECTORS

Eigenvalues and Eigenvectors

- Let $A \in M_{nn}(\mathbb{R})$ be a square matrix. We say that $\lambda \in \mathbb{C}$ is an eigenvalue of A with eigenvector $x \in M_{n1}(\mathbb{C})$ if and only if

$$Ax = \lambda x \quad .$$

with $x \neq 0$

- With every eigenvalue λ we associate an eigenvector space $E_\lambda(A)$ which consists of all vectors x that are eigenvectors to the eigenvalue λ. Thus:

$$E_\lambda(A) = \{ x \in M_{n1}(\mathbb{C}) \mid Ax = \lambda x \}$$

How to find the eigenvalues

- $\boxed{\lambda \text{ eigenvalue of } A \iff \det(A - \lambda I) = 0}$

Proof

Note that

$$Ax = \lambda x \Leftrightarrow Ax - \lambda x = 0 \Leftrightarrow Ax - \lambda I x = 0$$

$$\Leftrightarrow (A - \lambda I)x = 0 \qquad (1)$$

Eq. (1) has an obvious solution $x = 0$ and if $\det(A - \lambda I) \neq 0$, then this solution is unique. It follows that

λ eigenvalue of $A \Leftrightarrow$

$\Leftrightarrow$ Eq. (1) has a solution $x \neq 0$

$\Leftrightarrow$ $x = 0$ is NOT a unique solution

$\Leftrightarrow$ $\det(A - \lambda I) = 0$.

example

Find the eigenvalues of

$$A = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 3 & -1 \\ 2 & 1 & 3 \end{bmatrix}$$

$$\det(A - \lambda I) = \det\left(\begin{bmatrix} 2 & -1 & 1 \\ 0 & 3 & -1 \\ 2 & 1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) =$$

$$= \begin{vmatrix} 2-\lambda & -1 & 1 \\ 0 & 3-\lambda & -1 \\ 2 & 1 & 3-\lambda \end{vmatrix} =$$

$$\begin{array}{ccc} & \uparrow\!\underline{\qquad\quad} & 3-\lambda \end{array}$$

$$= \begin{vmatrix} 2-\lambda & 2-\lambda & 1 \\ 0 & 0 & -1 \\ 2 & (3-\lambda)^2+1 & 3-\lambda \end{vmatrix} \longrightarrow \; =$$

$$= -(-1)\begin{vmatrix} 2-\lambda & 2-\lambda \\ 2 & (3-\lambda)^2+1 \end{vmatrix} = (2-\lambda)\begin{vmatrix} 1 & 1 \\ 2 & (3-\lambda)^2+1 \end{vmatrix}$$

$$= (2-\lambda)\left[(3-\lambda)^2+1-2\right] = (2-\lambda)\left[(3-\lambda)^2-1\right]$$

$$= (2-\lambda)(3-\lambda-1)(3-\lambda+1) = (2-\lambda)(2-\lambda)(4-\lambda).$$

λ eigenvalue of $A \Leftrightarrow \det(A-\lambda I)=0 \Leftrightarrow$

$\Leftrightarrow (2-\lambda)(2-\lambda)(4-\lambda)=0$

$\Leftrightarrow \underline{\lambda = 2 \text{ or } \lambda = 4}$

$\lambda = 2$: <u>double</u> eigenvalue

How to find the eigenvectors

For each eigenvalue λ we use Gaussian Elimination to solve the equation

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

The solution space of this equation coincides with the eigenvector space $E_\lambda(A)$.

example

In the previous example:

For $\lambda=2$

$$M \sim \left[\begin{array}{ccc|c} 2-\lambda & -1 & 1 & 0 \\ 0 & 3-\lambda & -1 & 0 \\ 2 & 1 & 3-\lambda & 0 \end{array}\right] \sim$$

$$\sim \left[\begin{array}{ccc|c} 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 2 & 1 & 1 & 0 \end{array}\right]\begin{array}{c} \leftarrow \\ \\ \leftarrow \end{array} \sim$$

$$\sim \left[\begin{array}{ccc|c} 2 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & +1 & 0 \end{array}\right]\begin{array}{c} \\ (+1) \\ \hookleftarrow \end{array}$$

$$\sim \left[\begin{array}{ccc|c} 2 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \sim$$

$$\sim \left[\begin{array}{ccc|c} 2 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{array}\right] \begin{array}{l} \leftarrow\!\urcorner \\ (-1) \end{array} \sim$$

$$\sim \left[\begin{array}{ccc|c} 2 & 0 & 2 & 0 \\ 0 & 1 & -1 & 0 \end{array}\right] \cdot (1/2) \sim$$

$$\sim \left[\begin{array}{ccc|c} 1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{array}\right] \Leftrightarrow \begin{cases} x + z = 0 \\ y - z = 0 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} x = -z \\ y = z \end{cases} \Leftrightarrow (x, y, z) = (-z, z, z) = z(-1, 1, 1)$$

thus

$$E_2(A) = \left\{ z \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \;\middle|\; z \in \mathbb{R} \right\}$$

EXAMPLES – THEORETICAL

a) Let $A \in M_n(\mathbb{R})$ be a matrix with $A^2 = 5A - 6I$. Show that

λ eigenvalue of $A \Rightarrow \lambda = 2 \vee \lambda = 3$.

Solution

Let $\lambda \in \lambda(A)$ be an eigenvalue of A with x a corresponding eigenvector. It follows that

$Ax = \lambda x$ and

$A^2 x = (AA)x = A(Ax) = A(\lambda x) = \lambda(Ax) = \lambda(\lambda x) = \lambda^2 x.$

and therefore:

$(A^2 - 5A + 6I)x = \mathbf{0}x = \mathbf{0}$

$A^2 = 5A - 6I \Rightarrow A^2 - 5A + 6I = \mathbf{0} \Rightarrow$

$\Rightarrow (A^2 - 5A + 6I)x = \mathbf{0}x = \mathbf{0} \qquad (1)$

and $(A^2 - 5A + 6I)x = A^2x - 5Ax + 6Ix = \lambda^2 x - 5\lambda x + 6x =$

$= (\lambda^2 - 5\lambda + 6)x \qquad (2)$

From (1) and (2):

$(\lambda^2 - 5\lambda + 6)x = \mathbf{0} \Rightarrow \lambda^2 - 5\lambda + 6 = 0 \Rightarrow (\lambda - 2)(\lambda - 3) = 0$

$\Rightarrow \lambda = 2 \vee \lambda = 3$

EXERCISES

① Find the eigenvalues and corresponding eigenvector spaces for the following matrices.

a) $A = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 4 & -7 & 1 \end{bmatrix}$ $\qquad (\lambda = 1 \longrightarrow t(0,0,1))$

b) $B = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 3 & 3 & 20 \end{bmatrix}$ $\qquad \begin{matrix} (\lambda = 1 \longrightarrow t(1,-1,0) \\ \lambda = 2 \longrightarrow t(3,3,-1) \\ \lambda = 21 \longrightarrow t(1,1,6)) \end{matrix}$

c) $C = \begin{bmatrix} 5 & -6 & -6 \\ -1 & 4 & 2 \\ 3 & -6 & -4 \end{bmatrix}$ $\qquad \begin{matrix} (\lambda = 1 \longrightarrow t(3,-1,3) \\ \lambda = 2 \longrightarrow t(2,2,-1)) \end{matrix}$

[answers can be confirmed by matlab or octave]

(2) Let $A = \begin{bmatrix} 1 & 2a+1 \\ 2a-1 & 1 \end{bmatrix}$

For what values of a does A have only one eigenvalue?

(3) Rotation matrix

Let $R(\vartheta) = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}$

Show that $R(\vartheta)$ has real eigenvalues if and only if $\sin\vartheta = 0$.

(4) Find the eigenvalues of the following matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

(ans: $\lambda = +1, -1$)

(5) Let $A \in M_{nn}(\mathbb{R})$ be a matrix such that $A^2 = I$. Show that if λ is an eigenvalue of A, then $\lambda = 1$ or $\lambda = -1$.

(6) Let $A \in M_{nn}(\mathbb{R})$ be a non-singular matrix. Show that if $\lambda \neq 0$ is an eigenvalue of A, then $1/\lambda$ is an eigenvalue of A^{-1}.

⑦ Let $A \in M_n(\mathbb{R})$ with $A^2+3A=-2I$. Show that if:
λ eigenvalue of $A \Rightarrow \lambda=-1 \vee \lambda=-2$.

⑧ Let $A \in M_n(\mathbb{R})$ with $A^{-1}=2I-A$. Show that:
λ eigenvalue of $A \Rightarrow \lambda=1$.

▼ Characteristic polynomial

- The determinant $\det(A-\lambda I)$ simplifies to a polynomial of the form

$$\det(A-\lambda I) = (-1)^n \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1 \lambda + c_0$$

We call this polynomial the <u>characteristic polynomial</u> of A.

<u>Proof</u>

By definition

$$\det(A-\lambda I) = \sum_{\sigma\in S_n} s(\sigma) \prod_{a=1}^{n} (A-\lambda I)_{a,\sigma(a)}$$

For $\sigma_0(a)=a$ (the do-nothing permutation) we have $s(\sigma)=1$ and

$$\prod_{a=1}^{n} (A-\lambda I)_{a,\sigma(a)} = \prod_{a=1}^{n} (A-\lambda I)_{aa} =$$

$$= \prod_{a=1}^{n} (A_{aa}-\lambda) = (-1)^n \lambda^n + \cdots \text{ lower order terms.}$$

We also note that for $\sigma \neq \sigma_0$ the product contains at least one off-diagonal element, all of which are constant (i.e. independent of λ), so it does not contribute additional "λ^n" terms, but only lower-order terms. The conclusion follows. □

- From the fundamental theorem of algebra we know that the characteristic polynomial will <u>always</u> have n roots:

$$\lambda_1, \lambda_2, \dots, \lambda_n$$

therefore it can be factored as

$$\begin{aligned}\det(A-\lambda I) &= (-1)^n(\lambda-\lambda_1)(\lambda-\lambda_2)\cdots(\lambda-\lambda_n)\\ &= (\lambda_1-\lambda)(\lambda_2-\lambda)\cdots(\lambda_n-\lambda)\\ &= \prod_{a=1}^{n}(\lambda_a-\lambda).\end{aligned}$$

It follows that

$$\boxed{\det A = \lambda_1\lambda_2\lambda_3\cdots\lambda_n}$$

<u>Proof</u>

$$\det A = \det(A-0I) = \prod_{a=1}^{n}(\lambda_a-0) = \prod_{a=1}^{n}\lambda_a$$

$$= \lambda_1\lambda_2\lambda_3\cdots\lambda_n. \quad \square$$

Trace of a Matrix

- Let $A \in M_{nn}(\mathbb{R})$ be a square matrix. We define the <u>trace</u> of A as the sum of the diagonal elements of A:

$$\boxed{tr(A) = \sum_{a=1}^{n} A_{aa}}$$

- If $\lambda_1, \lambda_2, \dots, \lambda_n$ are the roots of the characteristic polynomial, then we can show that

$$\boxed{\lambda_1 + \lambda_2 + \cdots + \lambda_n = tr(A)}$$

Proof

Note that if

$$\begin{aligned} \det(A - \lambda I) &= (\lambda_1 - \lambda)(\lambda_2 - \lambda) \cdots (\lambda_n - \lambda) \\ &= (-1)^n \lambda^n + (-1)^n(\lambda_1 + \lambda_2 + \cdots + \lambda_n)\lambda^{n-1} \\ &\quad + \cdots + c_0 \end{aligned}$$

then $\underline{c_{n-1} = (-1)^n(\lambda_1 + \lambda_2 + \cdots + \lambda_n)}$.

Consider again the permutation $\sigma_0(a) = a$. By definition, we have again:

$$\det(A-\lambda I) = \sum_{\sigma\in S_n} s(\sigma)\prod_{a=1}^{n}(A-\lambda I)_{a,\sigma(a)} =$$

$$= \prod_{a=1}^{n}(A_{aa}-\lambda) + g(\lambda)$$

with

$$g(\lambda) = \sum_{\sigma\in S_n-\{\sigma_0\}} s(\sigma)\prod_{a=1}^{n}(A-\lambda I)_{a,\sigma(a)}$$

Note that any permutation $\sigma\neq\sigma_0$ has at least one transposition and thus knocks out at least two diagonal elements It follows that the degree of $g(\lambda)$ satisfies

$$\deg g(\lambda) < n-1$$

and thus $g(\lambda)$ does not contribute to the λ^{n-1} term. So all contributions to λ^{n-1} come from

$$\prod_{a=1}^{n}(A_{aa}-\lambda) = (-1)^n\prod_{a=1}^{n}(\lambda-A_{aa}) =$$

$$= (-1)\left[\lambda^n + \left(\sum_{a=1}^{n}A_{aa}\right)\lambda^{n-1} + \cdots\right]$$

which gives $\underline{c_{n-1} = (-1)^n(A_{11}+A_{22}+\cdots+A_{nn})}$ □

EXAMPLES

a) Let $A = \begin{bmatrix} a+3 & 1 \\ 2a & 2 \end{bmatrix}$ and let $\lambda_1, \lambda_2 \in \mathbb{C}$ be the eigenvalues of A. Find all $a \in \mathbb{R}$ such that $\frac{1}{\lambda_1^2} + \frac{1}{\lambda_2^2} = 1$

Solution

$$\lambda_1 \lambda_2 = \det A = \begin{vmatrix} a+3 & 1 \\ 2a & 2 \end{vmatrix} = 2(a+3) - 1 \cdot 2a = 2a+6-2a = 6$$

and

$$\lambda_1 + \lambda_2 = \operatorname{tr} A = (a+3) + 2 = a+5$$

so it follows that

$$\frac{1}{\lambda_1^2} + \frac{1}{\lambda_2^2} = \frac{\lambda_1^2 + \lambda_2^2}{\lambda_1^2 \lambda_2^2} = \frac{(\lambda_1 + \lambda_2)^2 - 2(\lambda_1 \lambda_2)}{(\lambda_1 \lambda_2)^2} =$$

$$= \frac{(\operatorname{tr} A)^2 - 2 \det A}{(\det A)^2} = \frac{(a+5)^2 - 2 \cdot 6}{6^2} =$$

$$= \frac{(a+5)^2 - 12}{36} .$$

and therefore:

$$\frac{1}{\lambda_1^2} + \frac{1}{\lambda_2^2} = 1 \Leftrightarrow \frac{(a+5)^2 - 12}{36} = 1 \Leftrightarrow (a+5)^2 - 12 = 36$$

$$\Leftrightarrow (a+5)^2 = 12 + 36 \Leftrightarrow (a+5)^2 = 48 = 16 \cdot 3 = 4^2 \cdot 3$$

$$\Leftrightarrow a+5 = 4\sqrt{3} \vee a+5 = -4\sqrt{3} \Leftrightarrow$$

$$\Leftrightarrow a = -5 + 4\sqrt{3} \vee a = -5 - 4\sqrt{3}$$

EXERCISES

(9) Let
$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2x+3 & 1 \\ 1 & 1 & 2x-1 \end{bmatrix}$$
If $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues of A, find $x \in \mathbb{C}$ such that

a) $\lambda_1 \lambda_2 \lambda_3 = 1$

b) $\lambda_1 + \lambda_2 + \lambda_3 = 4$

(10) Let $A = \begin{bmatrix} x+1 & 2x \\ 3 & x-1 \end{bmatrix}$

with eigenvalues λ_1, λ_2. Find $x \in \mathbb{C}$ such that

a) $\lambda_1 + \lambda_2 = 3$

b) $\dfrac{1}{\lambda_1} + \dfrac{1}{\lambda_2} = 1$

c) $\lambda_1^2 + \lambda_2^2 = 1$ (Hint: $a^2+b^2 = (a+b)^2 - 2ab$)

(11) Let $A = \begin{bmatrix} a & 1 \\ 1 & 1 \end{bmatrix}$

with eigenralues λ_1, λ_2. Find $a \in \mathbb{C}$ such that $\lambda_1^3 + \lambda_2^3 = 0$.

(Hint: $(a+b)^3 = (a^3+b^3) + 3ab(a+b)$)

↳ The following problems use
a) $\det(AB) = \det(A)\det(B)$
b) $\det(I) = 1$.

⑫ Let $A \in M_{nn}(\mathbb{R})$ be a matrix and let $B = P^{-1}AP$ with $P \in M_{nn}(\mathbb{R})$ a non-singular matrix. Show that A, B have the same eigenvalues.

⑬ Let $A, B \in M_{nn}(\mathbb{R})$ with A non-singular. Show that AB and BA have the same eigenvalues.

⑭ Let $A \in M_{22}(\mathbb{R})$ be a 2×2 matrix. If A is non-singular, show that

$$\operatorname{tr}(A^{-1}) = \frac{\operatorname{tr}(A)}{\det(A)}$$

↳ (see exercise ⑥)

• ⑮ Let $A \in M_{33}(\mathbb{R})$ be a 3×3 non-singular matrix with eigenvalues $\lambda_1, \lambda_2, \lambda_3$ that satisfy

$$\lambda_1^2 + \lambda_2^2 + \lambda_3^2 = 1$$

Show that

$$\operatorname{tr}(A^{-1}) = \frac{(\operatorname{tr}(A)+1)(\operatorname{tr}(A)-1)}{2\det A}$$

Cayley-Hamilton theorem

Thm: Let $A \in M_n(\mathbb{R})$ be a square matrix ~~wh~~ with characteristic polynomial

$$\det(A-\lambda I) = (-1)\lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$$

Then A satisfies

$$(-1)^n A^n + c_{n-1}A^{n-1} + \dots + c_1 A + c_0 I = \mathbf{0}$$

▸ **Method:** The Cayley-Hamilton theorem provides a second method for calculating the matrix inverse A^{-1} as shown in the following example

EXAMPLE

Given the matrix

$$A = \begin{bmatrix} 5 & 4 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 2 \end{bmatrix}$$

use the Cayley-Hamilton theorem to write A^{-1} in terms of A.

Solution

Since,

$$\det(A-\lambda I) = \begin{vmatrix} 5-\lambda & 4 & 0 \\ 1 & 2-\lambda & 0 \\ 1 & 2 & 2-\lambda \end{vmatrix} =$$

$$= (2-\lambda) \begin{vmatrix} 5-\lambda & 4 \\ 1 & 2-\lambda \end{vmatrix} =$$

$$= (2-\lambda)\left[(5-\lambda)(2-\lambda) - 4 \cdot 1\right] =$$

$$= (2-\lambda)(10 - 5\lambda - 2\lambda + \lambda^2 - 4) =$$

$$= (2-\lambda)(\lambda^2 - 7\lambda + 6) =$$

$$= 2\lambda^2 - 14\lambda + 12 - \lambda^3 + 7\lambda^2 - 6\lambda =$$

$$= -\lambda^3 + (2+7)\lambda^2 + (-14-6)\lambda + 12$$

$$= -\lambda^3 + 9\lambda^2 - 20\lambda + 12 \Rightarrow$$

$$\Rightarrow -A^3 + 9A^2 - 20A + 12I = 0 \Rightarrow$$

$$\Rightarrow A^3 - 9A^2 + 20A = 12I \Rightarrow$$

$$\Rightarrow A(A^2 - 9A + 20I) = 12I \Rightarrow$$

$$\Rightarrow A\left[\frac{1}{12}(A^2 - 9A + 20I)\right] = I \Rightarrow$$

$$\rightarrow A^{-1} = \frac{1}{12}(A^2 - 9A + 20I).$$

▸ Method: We can also use the Cayley-Hamilton theorem to write higher powers A^n in terms of a few powers of A.

EXAMPLE

Given the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Show that $\forall n \in \mathbb{N} - \{0,1\} : A^n = nA - (n-1)I$.

Solution

$$\det(A-\lambda I) = \begin{vmatrix} 1-\lambda & 0 \\ 1 & 1-\lambda \end{vmatrix} = (1-\lambda)^2 = \lambda^2-2\lambda+1 \Rightarrow$$

$\Rightarrow A^2-2A+I=0 \Rightarrow A^2=2A-I.$

Assume that for $n=k$: $A^k = kA-(k-1)A$

For $n=k+1$, we will show that: $A^{k+1}=(k+1)A-kI$

We have:

$$A^{k+1} = A^k A = [kA-(k-1)I]A =$$
$$= kA^2-(k-1)A = k(2A-I)-(k-1)A =$$
$$= 2kA-kI-kA+A = (2k-k+1)A-kI =$$
$$= (k+1)A-kI.$$

EXERCISES

16. For the following matrices, write A^{-1} and A^3 in terms of A and I.

a) $A = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}$ b) $A = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$

c) $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ d) $A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

17. For the following matrices, write A^{-1} and A^4 in terms of I, A, A^2.

a) $A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ b) $A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$

c) $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ d) $A = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

▼ Linear systems of differential equations

- Recall that the solution to the differential equation

$$\frac{dy(t)}{dt} = a\,y(t) + f(t)$$

with initial condition

$$y(0) = y_0$$

is given by:

$$y(t) = e^{at} y_0 + e^{at} \int_0^t e^{-a\tau} f(\tau)\,d\tau$$

- We want the solution to the more general system of linear differential equations:

$$\begin{cases} \dfrac{dy_1(t)}{dt} = A_{11}\,y_1(t) + A_{12}\,y_2(t) + \dots + A_{1n}\,y_n(t) + f_1(t) \\ \dfrac{dy_2(t)}{dt} = A_{21}\,y_1(t) + A_{22}\,y_2(t) + \dots + A_{2n}\,y_n(t) + f_2(t) \\ \quad\vdots \\ \dfrac{dy_n(t)}{dt} = A_{n1}\,y_1(t) + A_{n2}\,y_2(t) + \dots + A_{nn}\,y_n(t) + f_n(t) \end{cases}$$

with initial condition:

$$y_1(0) = b_1 \ , \ y_2(0) = b_2 , \ \dots , \ y_n(0) = b_n$$

If we define

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} , \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} , \quad y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix}$$

then the system can be rewritten as:

$$\boxed{\begin{cases} \dfrac{dy(t)}{dt} = A y(t) + f(t) \\ y(0) = b \end{cases}}$$

The solution to this system is based on the matrix exponential.

↕→ The matrix exponential

- Let $A \in M_{nn}(\mathbb{R})$ be a square matrix. The exponential of A is defined as

$$\boxed{\exp(A) = \sum_{n=0}^{+\infty} \frac{1}{n!} A^n}$$

with $0! = 1$

$n! = n(n-1)(n-2)\cdots 3\cdot 2\cdot 1$

This generalizes the identity

$$e^x = \sum_{n=0}^{+\infty} \frac{x^n}{n!}$$

- The matrix exponential $\exp(A)$ converges for all matrices A.

▶ Properties

a) $\frac{d}{dt}\exp(tA) = A\exp(tA) = \exp(tA)A$

b) $[\exp(A)]^{-1} = \exp(-A)$

c) $AB = BA \Rightarrow \exp(A+B) = \exp(A)\exp(B)$

d) $\frac{dy(t)}{dt} = Ay(t) \rightarrow y(t) = \exp(tA)y(0)$

e) $\exp((t_1+t_2)A) = \exp(t_1 A)\exp(t_2 A)$

Solution to linear system of ODEs

The solution to the ODE system

$$\frac{dy(t)}{dt} = Ay(t) + f(t)$$

is given by

$$\boxed{y(t) = \exp(tA)\,y(0) + \exp(tA)\int_0^t \exp(-\tau A) f(\tau)\,d\tau}$$

How to calculate the matrix exponential

To calculate $\exp(tA)$ we work as follows:

•1 Let $A \in M_{nn}(\mathbb{R})$. From the Cayley-Hamilton theorem we conclude that there are coefficients $c_0, c_1, \dots, c_{n-1}$ such that

$$\exp(tA) = c_{n-1} A^{n-1} t^{n-1} + \dots + a_1 At + a_0 I$$

Before we find $c_0, \dots, c_{n-1}$ we simplify the right-hand side of the expression above.

•2 We find the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ of the matrix A.

•3 Let
$f(x) = c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$
To find the coefficients $c_0, c_1, \ldots, c_{n-1}$:
a) If λ_k is an eigenvalue of $\underline{\underline{tA}}$, then

$$e^{\lambda_k} = f(\lambda_k)$$

b) If λ_k is an eigenvalue of A with multiplicity m, then we also have:

$$\begin{cases} e^{\lambda_k} = f'(\lambda_k) \\ e^{\lambda_k} = f''(\lambda_k) \\ \quad\vdots \\ e^{\lambda_k} = f^{(m-1)}(\lambda_k) \end{cases}$$

Thus we get a system of n equations from which we find $c_0, \ldots, c_{n-1}$.

•4 knowing the coefficients $c_0, \ldots, c_{n-1}$ we now calculate the exponential $\exp(tA)$.

example

For $A = \begin{bmatrix} 1 & 1 \\ 9 & 1 \end{bmatrix}$

we have

$$\exp(tA) = c_1 t A + c_0 I = c_1 t \begin{bmatrix} 1 & 1 \\ 9 & 1 \end{bmatrix} + c_0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_1 t + c_0 & c_1 t \\ 9 c_1 t & c_1 t + c_0 \end{bmatrix}$$

- Eigenvalues of A:

$$\det(A - \lambda I) = \begin{vmatrix} 1-\lambda & 1 \\ 9 & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - 9 =$$

$$= (1-\lambda-3)(1-\lambda+3) =$$

$$= (-\lambda-2)(-\lambda+4) =$$

$$= (\lambda+2)(\lambda-4) \Rightarrow$$

$\Rightarrow$ The eigenvalues of A are $\lambda_1 = -2$ and $\lambda_2 = 4$

$\Rightarrow$ The eigenvalues of tA are $\lambda_1 = -2t$ and $\lambda_2 = 4t \Rightarrow$

$$\Rightarrow \begin{cases} e^{4t} = c_1(4t) + c_0 \\ e^{-2t} = c_1(-2t) + c_0 \end{cases} \Leftrightarrow \dots \Leftrightarrow \begin{cases} c_1 = \frac{1}{6t}(e^{4t} - e^{-2t}) \\ c_0 = \frac{1}{3}(e^{4t} + 2e^{-2t}) \end{cases}$$

It follows that

$$\exp(tA) = \cdots = \frac{1}{6}\begin{bmatrix} 3e^{4t}+3e^{-2t} & e^{4t}-e^{-2t} \\ 9e^{4t}-9e^{-2t} & 3e^{4t}+3e^{-2t} \end{bmatrix}$$

example

For $A = \begin{bmatrix} 0 & 1 \\ -9 & 6 \end{bmatrix}$

we have

$$\exp(tA) = c_1 t A + c_0 I = \cdots$$

$$= \begin{bmatrix} c_0 & c_1 t \\ -9c_1 t & 6c_1 t + c_0 \end{bmatrix}$$

Eigenvalues:

$$\det(A-\lambda I) = \begin{vmatrix} -\lambda & 1 \\ -9 & 6-\lambda \end{vmatrix} = -\lambda(6-\lambda) - (-9) =$$

$$= -6\lambda + \lambda^2 + 9 = (\lambda-3)^2 \Rightarrow$$

$\Rightarrow \lambda = 3$ double eigenvalue of A

$\Rightarrow \lambda = 3t$ double eigenvalue of tA

For $f(x) = c_1 x + c_0 \Rightarrow f'(x) = c_1$ thus

$$\begin{cases} e^{3t} = c_1(3t) + c_0 \\ e^{3t} = c_1 \end{cases} \Leftrightarrow \dots \Leftrightarrow \begin{cases} c_0 = e^{3t}(1-3t) \\ c_1 = e^{3t} \end{cases}$$

thus

$$\exp(tA) = \dots = \begin{bmatrix} (1-3t)e^{3t} & te^{3t} \\ -9te^{3t} & (1+3t)e^{3t} \end{bmatrix}$$

Now let us consider

$$\begin{cases} \dfrac{dy_1}{dt} = y_2 \\ \dfrac{dy_2}{dt} = -9y_1 + 6y_2 \end{cases} \Leftrightarrow \frac{d}{dt}\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -9 & 6 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

It follows that

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \exp(tA)\begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} =$$

$$= \begin{bmatrix} (1-3t)e^{3t} & te^{3t} \\ -9te^{3t} & (1+3t)e^{3t} \end{bmatrix}\begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{cases} y_1(t) = (1-3t)e^{3t}y_1(0) + te^{3t}y_2(0) \\ y_2(t) = -9te^{3t}y_1(0) + (1+3t)e^{3t}y_2(0) \end{cases}$$

2nd method

For a 2×2 matrix A with eigenvalues λ_1, λ_2, the matrix exponential is given by:

a) If $\lambda_1 \neq \lambda_2$ then

$$\boxed{\exp(tA) = \frac{\lambda_1 e^{\lambda_2 t} - \lambda_2 e^{\lambda_1 t}}{\lambda_1 - \lambda_2} I + \frac{e^{\lambda_1 t} - e^{\lambda_2 t}}{\lambda_1 - \lambda_2} A}$$

b) If $\lambda_1 = \lambda_2 = \lambda$ then

$$\boxed{\exp(tA) = e^{\lambda t}(1 - \lambda t) I + t e^{\lambda t} A}$$

EXERCISES

(18) Use the matrix exponential to solve the following systems in terms of $y_1(0)$ and $y_2(0)$:

a) $$\begin{cases} dy_1/dt = 4y_1 + y_2 \\ dy_2/dt = -2y_1 + y_2 \end{cases}$$

b) $$\begin{cases} dy_1/dt = -5y_1 - y_2 \\ dy_2/dt = y_1 - 3y_2 \end{cases}$$

c) $$\begin{cases} dy_1/dt = y_1 \\ dy_2/dt \quad y_1 + y_2 \end{cases}$$

(19) Rotation matrix.
Show that the rotation matrix

$$R(\vartheta) = \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix}$$

satisfies:

$$R(\vartheta) = \exp\left[\vartheta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right], \quad \vartheta \in \mathbb{R}.$$

↳ Use $e^{i\vartheta} = \cos\vartheta + i\sin\vartheta$.

GRAPH THEORY II

Adjacency matrix

Def : Let G be a graph with vertices $V(G)=\{u_1,u_2,\dots,u_n\}$.

a) We define the adjacency matrix $A(G)\in M_n(\mathbb{R})$ via

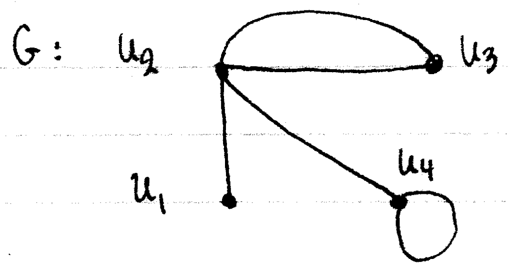
$$\forall a,b\in[n]: [A(G)]_{ab} = \begin{cases} |\{e\in E(G) \mid \psi_G(e)=\{u_a,u_b\}\}| , & \text{if } a\neq b \\ 2|\{e\in E(G) \mid \psi_G(e)=\{u_a\}\}| , & \text{if } a=b \end{cases}$$

b) We define the connectivity matrix $B(G)\in M_n(\mathbb{R})$ via

$$B(G) = A(G) + A^2(G) + \dots + A^{n-1}(G)$$

EXAMPLE

Consider the graph G given by



The corresponding adjacency matrix is:

$$A(G) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}$$

↳ Note that loops are counted twice.

Properties of the adjacency matrix

① → $A(G), B(G)$ are both symmetric

$$\boxed{\begin{array}{l}\forall a,b\in[n]: [A(G)]_{ab} = [A(G)]_{ba} \\ \forall a,b\in[n]: [B(G)]_{ab} = [B(G)]_{ba}\end{array}}$$

② → Column / Row sums of $A(G)$

$$\boxed{\begin{array}{l}\forall b\in[n]: \sum_{a=1}^{n} [A(G)]_{ab} = d(u_b) \\ \forall a\in[n]: \sum_{b=1}^{n} [A(G)]_{ab} = d(u_a)\end{array}}$$

Enumeration of walks

notation: Recall that $W(G)$ is the set of all walks on the graph G. Let $a\in\mathbb{N}^*$. The set of all walks with length a will be denoted as:

$$W_a(G) = \{w\in W(G) \mid \ell(w) = a\}$$

The following result counts the number of walks of fixed length between two vertices.

Thm: Let G be a graph and let $u_a, u_b \in V(G)$ be two vertices, and let $k\in\mathbb{N}^*$. Then,

$$|\{w\in W_k(G) \mid s(w) = u_a \wedge t(w) = u_b\}| = [A^k(G)]_{ab}$$

Remark

a) A closed walk $w \in W(G)$ is a walk whose starting point $s(w)$ and endpoint $t(w)$ coincide

It follows from the previous result that the number of closed walks with length $k \in \mathbb{N}^*$ is given by:

$$\boxed{|\{w \in W_k(G) \mid s(w) = t(w)\}| = tr\left(A^k(G)\right)}$$

Note that if $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of $A(G)$, then $\lambda_1^k, \lambda_2^k, \dots, \lambda_n^k$ are the eigenvalues of $A^k(G)$. We may then conclude that

$$\boxed{tr\left(A^k(G)\right) = \lambda_1^k + \lambda_2^k + \dots + \lambda_n^k}$$

Consequently, it is easy to enumerate the number of closed walks, as a function of the length k, via the eigenvalues of the adjacency matrix for any graph G.

→ Graph connectivity

Thm : Let G be a graph with connectivity matrix $B(G)$. It follows that

G connected $\Leftrightarrow \forall a, b \in [n] : \left(a \neq b \Rightarrow [B(G)]_{ab} > 0\right)$

This theorem can be used algorithmically to calculate the number of components $w(G)$ for any graph as follows.

(a) Pick a vertex $u_a \in V(G)$

(b) Find all vertices $u_b \in V(G)$ such that $[B(G)]_{ab} > 0$. This defines a set of vertices V_0 such that $G[V_0]$ is a component of the graph G.

(c) Apply the same algorithm recursively to the graph $G - V_0$ to extract the next component.

(d) Upon removing all vertices, we will have found all graph components, and counting them will give us $w(G)$.

Given an algorithm that calculates $w(G)$, it is easy to write algorithms to calculate $\kappa(G)$ and $\lambda(G)$.

EXAMPLE

Let $A(K_4)$ be the adjacency matrix of K_4.

a) Find the characteristic polynomial of $A(K_4)$ and its eigenvalues

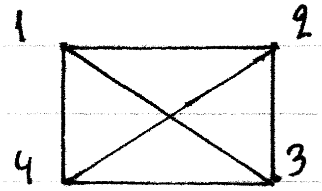
b) How many closed walks of length n does K_4 have?

c) Show that $A^4(K_4) = 6A^2(K_2) + 8A(K_4) + 3I$

d) How many open walks of length 4 does K_4 have?

Solution

a) For K_4:



the adjacency matrix is given by:

$$A(K_4) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \Rightarrow$$

$$\Rightarrow \det(A(K_4) - \lambda I) = \begin{vmatrix} -\lambda & 1 & 1 & 1 \\ 1 & -\lambda & 1 & 1 \\ 1 & 1 & -\lambda & 1 \\ 1 & 1 & 1 & -\lambda \end{vmatrix} \begin{matrix} \\ \\ \\ (-1) \quad \lambda \end{matrix} =$$

$$= \begin{vmatrix} 0 & \lambda+1 & \lambda+1 & 1-\lambda^2 \\ 0 & -\lambda-1 & 0 & \lambda+1 \\ 0 & 0 & -\lambda-1 & \lambda+1 \\ 1 & 1 & 1 & -\lambda \end{vmatrix} = -1 \begin{vmatrix} \lambda+1 & \lambda+1 & (\lambda+1)(1-\lambda) \\ -(\lambda+1) & 0 & \lambda+1 \\ 0 & -(\lambda+1) & \lambda+1 \end{vmatrix}$$

$\downarrow$

$$= (\lambda+1)^3 \begin{vmatrix} 1 & 1 & 1-\lambda \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{vmatrix} \begin{matrix} \urcorner \\ \hookleftarrow \\ \end{matrix} = (\lambda+1)^3 \begin{vmatrix} 1 & 1 & 1-\lambda \\ 0 & 1 & 2-\lambda \\ 0 & -1 & 1 \end{vmatrix} =$$

$$= (\lambda+1)^3 \begin{vmatrix} 1 & 2-\lambda \\ -1 & 1 \end{vmatrix} = (\lambda+1)^3 [1 \cdot 1 - (-1)(2-\lambda)] =$$

$$= (\lambda+1)^3 (1+2-\lambda) = (\lambda+1)^3 (3-\lambda)$$

and therefore

λ eigenvalue of $K_4 \Leftrightarrow \det(A(K_4) - \lambda I) = 0 \Leftrightarrow$

$\Leftrightarrow (\lambda+1)^3 \; (3-\lambda) = 0 \Leftrightarrow \lambda+1 = 0 \vee 3-\lambda = 0 \Leftrightarrow$

$\Leftrightarrow \lambda = -1 \vee \lambda = 3.$

b) The number N of closed walks with length n is given by:

$$N = |\{w \in W_n(K_4) \mid s(w) = t(w)\} =$$
$$= tr(A^n(K_4)) = \lambda_1^n + \lambda_2^n + \lambda_3^n + \lambda_4^n =$$
$$= (-1)^n + (-1)^n + (-1)^n + 3^n =$$
$$= 3(-1)^n + 3^n$$

c) Since

$$\det(A(K_4) - \lambda I) = (\lambda+1)^3(3-\lambda) = (\lambda^3 + 3\lambda^2 + 3\lambda + 1)(3-\lambda) =$$
$$= 3\lambda^3 + 9\lambda^2 + 9\lambda + 3 - \lambda^4 - 3\lambda^3 - 3\lambda^2 - \lambda =$$
$$= -\lambda^4 + (9-3)\lambda^2 + (9-1)\lambda + 3$$
$$= -\lambda^4 + 6\lambda^2 + 8\lambda + 3 \Rightarrow$$

$$\Rightarrow -A^4(K_4) + 6A^2(K_4) + 8A(K_4) + 3I = 0$$

$$\rightarrow A^4(K_4) = 6A^2(K_4) + 8A(K_4) + 3I$$

d)

$$A^2(K_4) = A(K_4)A(K_4) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix} \Rightarrow$$

$$\Rightarrow A^4(K_4) = 6A^2(K_4) + 8A(K_4) + 3I =$$

$$= 6\begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix} + 8\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} + 3\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 21 & 20 & 20 & 20 \\ 20 & 21 & 20 & 20 \\ 20 & 20 & 21 & 20 \\ 20 & 20 & 20 & 21 \end{bmatrix}$$

and therefore the number of walks of length 4 are given by:

$$N = |\{w \in W_4(K_4) \mid s(w) \neq t(w)\}| =$$

$$= \sum_{\substack{a,b \in [4] \\ a \neq b}} [A^4(K_4)]_{ab} = 20 \cdot 12 = 240$$

EXERCISES

(42) Write the incidence matrices for the following graphs:

a) K_3 d) $K_{3,3}$
b) K_4 e) C_4
c) $K_{2,3}$ f) P_4

(43) Let $A(K_3)$ be the adjacency matrix of K_3

a) Find the characteristic polynomial of $A(K_3)$

b) Show that $A^3 = 3A + 2I$

c) How many open walks does K_3 have of length 3?

d) How many closed walks does K_3 have of length 3?

e) Calculate A^5 and answer the same questions (c), (d) for walks of length 5.

(44) Let $A(K_{2,2})$ be the adjacency matrix of $K_{2,2}$

a) Find the characteristic polynomial of $A(K_{2,2})$

b) How many cycles of length 5 are there in $K_{2,2}$?

(45) Let G be a graph with adjacency matrix

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Show that G is not Eulerian.

(46) Show the following statements:

a) $$\sum_{a=1}^{n} \sum_{b=1}^{n} [A(G)]_{ab} = 2|E(G)|$$

b) $$\left. \begin{array}{l} \displaystyle\sum_{a=1}^{n} [A(G)]_{ab} \geqslant \frac{n}{2} \\ \text{for all } b \in [n] \end{array} \right\} \Rightarrow G \text{ Hamiltonian}$$

c) $$\sum_{a=1}^{n} [A(G)]_{ab} \geqslant \lambda(G) \ , \ \forall b \in [n]$$

▼ The shortest path problem

Def : A weighted graph G is a graph endowed with a mapping

$$f: E(G) \longrightarrow \mathbb{R}$$

which maps every edge $e \in E(G)$ to a unique real number $f(e)$.

Def : Let G be a weighted graph with $f: E(G) \to \mathbb{R}$ and let $w \in P(G)$ be a path on G. We define the weight $f(w)$ of the path w as:

$$\forall w \in P(G): f(w) = \sum_{e \in E(w)} f(e)$$

↳ Recall that a path $w \in P(G)$ is a walk on G where vertices and edges are visited no more than one time. $E(w)$ is the set of edges visited by w.

Def: Let G be a weighted graph with $f: E(G) \to \mathbb{R}$ and let $a, b \in V(G)$ with $a \neq b$ be two vertices of G. We define the distance $f(a,b)$ between a and b as:

$$f(a,b) = \min\{f(w) \mid w \in P(G, a \to b)\}$$

- The *shortest path problem* is, given a weighted graph G with $f: E(G) \to \mathbb{R}$ and vertices $a, b \in V(G)$ to find a path $w \in P(G, a \to b)$ such that $f(w) = f(a, b)$. Equivalently, we are looking for the path $w \in P(G, a \to b)$ that minimizes $f(w)$.

→ The Dijkstra algorithm

The Dijkstra algorithm can be used to solve the shortest path problem for the special case:

$\forall e \in E(G): f(e) > 0$

where all edges have a positive weight. The algorithm consists of two parts:

(1) *Preprocessing*: Given the initial vertex $a \in V(G)$, we define a mapping $L_k : V(G) \to \mathbb{R} \cup \{\infty\}$ with $k \in [n]$ and $n = |V(G)| - 1$ and ∞ representing a "fictional" infinite number. We also define a vertex sequence $u_0, u_1, \ldots, u_{n-1}$.

(2) *Postprocessing*: Given the terminal vertex $b \in V(G)$ and the output of the preprocessing step, the second step of the algorithm will find one (or all) shortest path from the initial vertex a to the terminal vertex b.

- Note that the postprocessing step can be repeated for different choices of a terminal vertex $b \in V(G)$ without repeating the preprocessing step. The preprocessing step needs to be repeated only if we change the initial vertex.

① → Preprocessing step

Let $a \in V(G)$ be the chosen initial vertex.

•1 We define

$$\begin{cases} u_0 = a \wedge \mathcal{S}_0 = \{u_0\} = \{a\} \\ \forall u \in V(G): L_0(u) = \begin{cases} 0 & , \text{ if } u = u_0 \\ \infty & , \text{ if } u \neq u_0 \end{cases} \end{cases}$$

•2 Assume that

$$\begin{cases} \mathcal{S}_K = \{u_0, u_1, \dots, u_K\} \\ L_K : V(G) \rightarrow \mathbb{R} \cup \{\infty\} \end{cases}$$

have already been defined.

▸ If $K = |V(G)| - 1$, then the algorithm terminates.

▸ Otherwise, we define L_{K+1} as follows:

1) $\forall u \in \mathcal{S}_K : L_{K+1}(u) = L_K(u)$

2) $\forall u \in V(G) - \mathcal{S}_K : (u, u_K \text{ not adjacent} \Rightarrow L_{K+1}(u) = L_K(u))$

3) $\forall u \in V(G) - \mathcal{S}_K : (u, u_K \text{ adjacent} \Rightarrow$

$$\Rightarrow L_{K+1}(u) = \min\{L_K(u), L_K(u_K) + f(uu_K)\}$$

with uu_K the edge between u and u_K.

▸ We define $u_{K+1} \in V(G)$ such that it satisfies:

$$\begin{cases} u_{K+1} \in V(G) - \mathcal{S}_K \\ \forall u \in V(G): L_{K+1}(u) \geq L_{K+1}(u_{K+1}) \end{cases}$$

Note that if multiple choices exist for u_{K+1}, we can fork the execution of the algorithm and pursue each choice. That may lead to multiple shortest paths of equal weight, if they exist.

▸ We define $\mathcal{S}_{K+1} = \mathcal{S}_K \cup \{u_{K+1}\}$.

A practical implementation of the algorithm is based on a grid of the form:

	$K=0$	$K=1$	$K=2$	$\cdots$	$K=n$
a	0	$L_1(a)$	$L_2(a)$	$\cdots$	$L_n(a)$
a_1	∞	$L_1(a_1)$	$L_2(a_1)$	$\cdots$	$L_n(a_1)$
a_2	∞	$L_1(a_2)$	$L_2(a_2)$	$\cdots$	$L_n(a_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_n	∞	$L_1(d_n)$	$L_2(a_n)$	$\cdots$	$L_n(a_n)$
		$u_0=a$	u_1	$\cdots$	u_{n-1}

We note that

1) The first column is initialized with 0 for the initial vertex a and with ∞ for all other vertices. We insert $u_0=a$ on column $K=1$.

2) Given the construction of column k, we define column $k+1$ as follows:

   - We copy from column k to column $k+1$ the numbers corresponding to the vertices $S_K=\{u_0, u_1, \ldots, u_K\}$, and we circle them.
   - We also copy any additional vertices that are not adjacent to u_K from among the set $V(G)-S_K$.
   - We use the formula
     $$L_{k+1}(u)=\min\{L_K(u), L_K(u_K)+f(uu_K)\}$$
     for any remaining vertices.

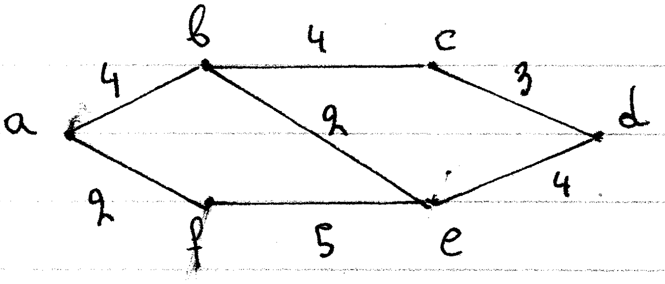
3) Given the column $k+1$, we choose from among the non-circled vertices (i.e. all $u \in \bar{V}(G) - \wp_k$) the one that minimizes $L_{k+1}(u)$. This defines u_{k+1}, which we write under the column $k+2$.

4) We define $\wp_{k+1} = \wp_k \cup \{u_k\}$.

EXAMPLE

Consider the following graph with initial vertex $a \in V(G)$:

G:



	K=0	k=1	k=2	k=3	k=4	K=5
a	0	⓪	⓪	⓪	⓪	⓪
b	∞	4	$\boxed{4}$	④	④	④
c	∞	∞	∞	8	$\boxed{8}$	⑧
d	∞	∞	∞	∞	10	10
e	∞	∞	7	$\boxed{6}$	⑥	⑥
f	∞	$\boxed{2}$	②	②	②	②
		a	f	b	e	c

Algorithm log:

$u_0 = a$

$S_0 = \{a\}$

For $K = 1$:

copy a with circle

copy c, d, e

$L_1(b) = \min\{L_0(b), L_0(a) + f(ab)\} = \min\{\infty, 0+4\} = 4$

$L_2(f) = \min\{L_0(f), L_0(a) + f(af)\} = \min\{\infty, 0+2\} = 2$

minimum at $f \longrightarrow u_1 = f$ and $S_1 = \{a, f\}$

For $k=2$:

copy a, f with circle

copy b, c, d

$L_2(e) = \min\{L_1(e), L_1(f) + f(ef)\} = \min\{\infty, 2+5\} = 7$

minimum at b → $u_2 = b$ and $S_2 = \{a, f, b\}$

For $k=3$:

copy a, f, b with circle

copy d

$L_3(c) = \min\{L_2(c), L_2(b) + f(bc)\} = \min\{\infty, 4+4\} = 8$

$L_3(e) = \min\{L_2(e), L_2(b) + f(be)\} = \min\{7, 4+2\} = 6$

minimum at e → $u_3 = e$ and $S_3 = \{a, f, b, e\}$

For $k=4$:

copy a, f, b, e with circle

copy c

$L_4(d) = \min\{L_3(d), L_3(e) + f(ed)\} = \min\{\infty, 6+4\} = 10$

minimum at c → $u_4 = c$ and $S_4 = \{a, f, b, e, c\}$

For $k=5$:

copy a, f, b, e, c with circle.

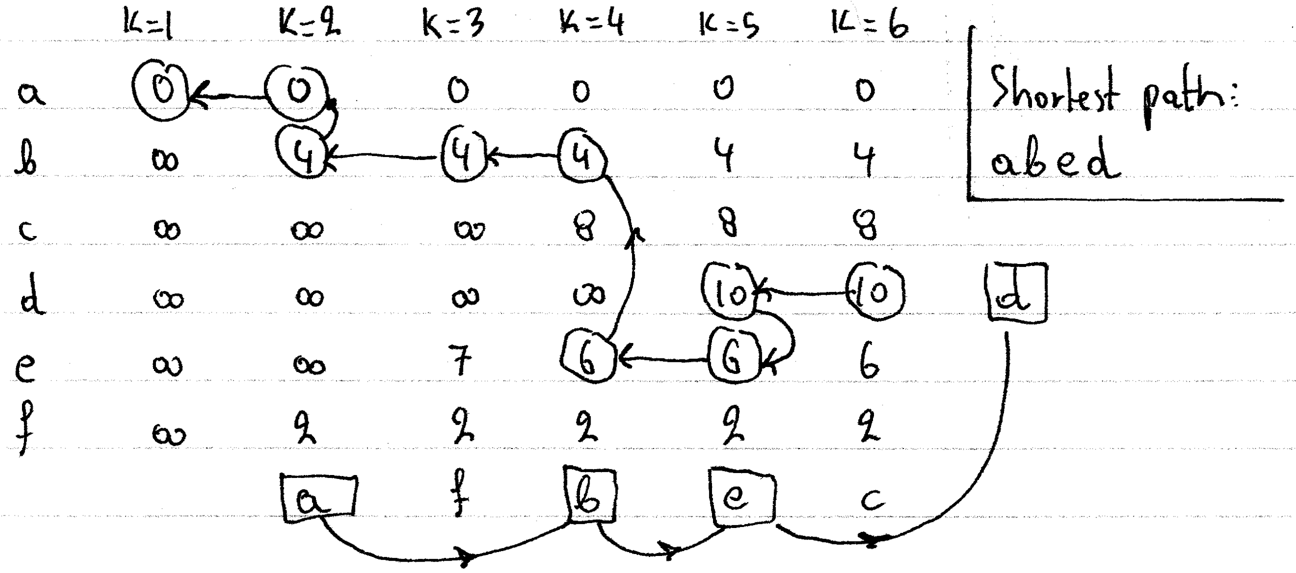
$L_5(d) = \min\{L_4(d), L_4(c) + f(cd)\} = \min\{10, 8+3\} = 10$

done!

↳ Note that L_5 gives the distance between the initial vertex a and all other vertices.
e.g. $f(a,c) = L_5(c) = 8$, etc...

② → Postprocessing

The postprocessing algorithm is illustrated by the following example where we seek the shortest path from vertex "a" to vertex "d" in the previous graph G:



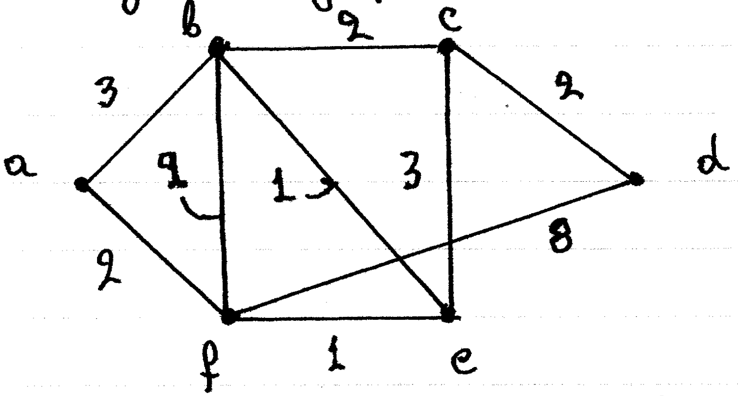
	K=1	K=2	K=3	K=4	K=5	K=6	
a	0	0	0	0	0	0	
b	∞	4	4	4	4	4	
c	∞	∞	∞	8	8	8	
d	∞	∞	∞	∞	10	10	d
e	∞	∞	7	6	6	6	
f	∞	2	2	2	2	2	
		a	f	b	e	c	

Shortest path: abed

- 1 On the last column we circle the number corresponding to the terminal vertex, which we show on the side
- 2 We move left, if the number to the left is equal. Otherwise, we square the vertex under the current column and move to the number corresponding to that vertex. Either way, we circle the next number.
- 3 Repeat until we get to column K=1.
- 4 The squared vertices form the shortest path

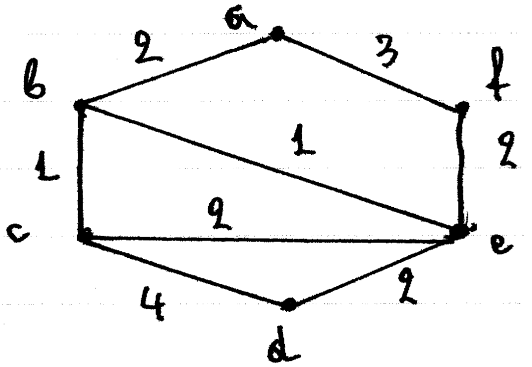
EXERCISES

(47) Apply Dijksta's algorithm to the following weighted graph.



Find the shortest path from a to d.

(48) Similarly, for the following graph



Find the shortest path from a to d.

▼ Tree graphs

To give a rigorous definition of trees, we need the following preliminary definitions first.

● → Preliminaries

Def: Let $f: A \to B$ be a mapping and let $S \subseteq A$ be given. We define

$$f(S) = \{f(x) \mid x \in S\}$$

or equivalently:

$$y \in f(S) \Leftrightarrow \exists x \in S : f(x) = y$$

We also say that

a) f one-to-one $\Leftrightarrow \forall x_1, x_2 \in A : (f(x_1) = f(x_2) \Rightarrow x_1 = x_2)$

b) f onto $\Leftrightarrow f(A) = B$

c) f bijection $\Leftrightarrow \begin{cases} f \text{ one-to-one} \\ f \text{ onto.} \end{cases}$

Def: (Isomorphic graphs)

Let G, H be two graphs. We say that $G \cong H$ (G isomorphic to H) if and only if there exist mappings

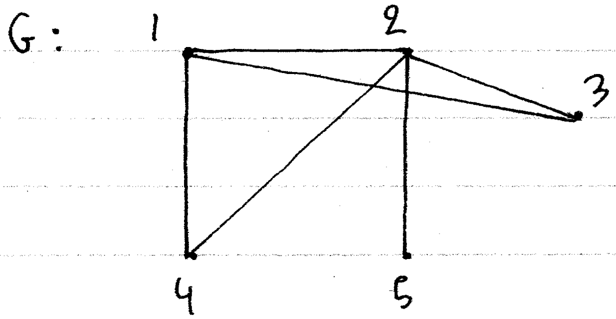
$$f: V(G) \to V(H) \quad \text{and} \quad g: E(G) \to E(H)$$

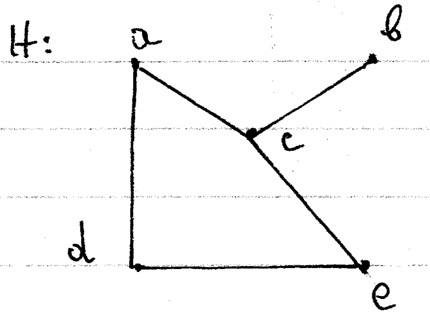
such that

$$\begin{cases} f, g \text{ are bijections} \\ \forall e \in E(G) : \psi_H(g(e)) = f(\psi_G(e)) \end{cases}$$

interpretation : The bijections f, g represent a relabelling of the vertices and edges of G. We say that $G \cong H$ when after such a relabelling, the vertices are connected by the edges the same way.

EXAMPLE





$G \cong H$ with bijections

$f(1) = d$ (both have degree 3)

$f(2) = c$ (both have degree 4)

$f(3) = a$ (3 connected with 1,2)

$f(4) = e$ (4 also connected with 1,2)

$f(5) = b$ (both have degree 1)

and

$g(12) = dc$ $\qquad$ $g(14) = de$

$g(23) = ca$ $\qquad$ $g(24) = ce$

$g(31) = ad$ $\qquad$ $g(25) = bc$

Def : (Subgraphs)

Let G, H be two graphs. We say that

$$H \subseteq G \Leftrightarrow \begin{cases} V(H) \subseteq V(G) \\ E(H) \subseteq E(G) \\ \forall e \in E(H) : \psi_H(e) = \psi_G(e) \end{cases}$$

notation: We denote the set of all subgraphs of a graph G as $\mathcal{P}(G)$ such that

$$H \in \mathcal{P}(G) \Leftrightarrow H \subseteq G$$

Definition of trees

A tree is defined as a connected acyclic graph. In detail, this is done via the following definitions:

Def: Let G be a graph. We say that

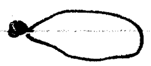
a) G cyclic $\Leftrightarrow \exists H \in \mathcal{P}(G) : \exists n \in \mathbb{N}^* : H \cong C_n$

b) G acyclic $\Leftrightarrow$ G not cyclic

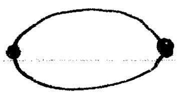
$\Leftrightarrow \forall H \in \mathcal{P}(G) : \forall n \in \mathbb{N}^* : H \ncong C_n$

Intuitively, G is cyclic if and only if there is a subgraph H of G such that it is isomorphic to the cycle graph C_n for some n. Recall that:

C_1 :



C_2:



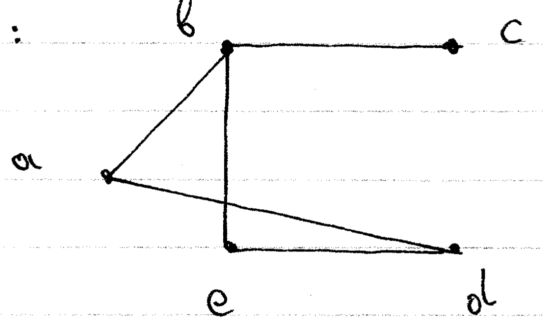
therefore loops and multiple edges automatically form cycles. It follows that:

$$\boxed{G \text{ acyclic} \Rightarrow G \text{ simple}}$$

EXAMPLE

Consider the graph

G :



Since

$G[\{a,b,d,e\}] \cong C_4 \Rightarrow G$ cyclic.

Def : Let G be a graph. We say that

$$G \text{ \underline{tree}} \Leftrightarrow \begin{cases} G \text{ connected} \\ G \text{ acyclic} \end{cases}$$

→ Properties of trees – Summary

(1) → $\boxed{G \text{ tree} \Rightarrow G \text{ simple}}$

(2) → A graph is a tree if and only if for any two distinct vertices there is a unique path that connects them

Thm: Let G be a graph. Then,
$$G \text{ tree} \Leftrightarrow \forall a,b \in V(G) : (a \neq b \Rightarrow |P(G, a \to b)| = 1)$$

(3) → A tree always disconnects to two components upon removing any one edge.

Thm: Let G be a graph. Then,
$$G \text{ tree} \Rightarrow \forall e \in E(G) : \omega(G - \{e\}) = 2$$

(4) → Relation between number of edges and vertices

Thm: Let G be a graph. Then,
$$G \text{ tree} \Leftrightarrow \begin{cases} |E(G)| = |V(G)| - 1 \\ G \text{ connected} \end{cases}$$

(5) → Trees and connectivity

Thm: Let G be a graph. Then,
$$G \text{ tree} \Rightarrow \kappa(G) = \lambda(G) = \delta(G) = 1$$

● → Properties of trees

(1) → $\boxed{G \text{ tree} \Rightarrow G \text{ simple}}$

Proof

$$\begin{aligned} G\text{ tree} &\Rightarrow \left\{\begin{array}{l} G\text{ acyclic}\\ G\text{ connected}\end{array}\right. \Rightarrow \\ &\Rightarrow G\text{ acyclic}\\ &\Rightarrow G\text{ simple}. \qquad \square\end{aligned}$$

(2) → A graph G is a tree if and only if for any two distinct vertices there is a unique path that connects them.

$$\boxed{\begin{array}{l}\underline{\text{Thm}}: \text{ Let } G \text{ be a graph. Then}\\ \quad G \text{ tree} \Leftrightarrow \forall a,b\in V(G) : (a\neq b \Rightarrow |P(G, a\to b)| = 1)\end{array}}$$

Proof

($\Rightarrow$): Assume that <u>G is a tree</u>. Let <u>$a,b\in V(G)$</u> be given and assume that <u>$a\neq b$</u>. To show that $|P(G,a\to b)|=1$ we assume that $|P(G,a\to b)|\neq 1$ and derive a contradiction. We distinguish between the following cases:

<u>Case 1</u>: Assume that $|P(G,a\to b)|=0$. Then,

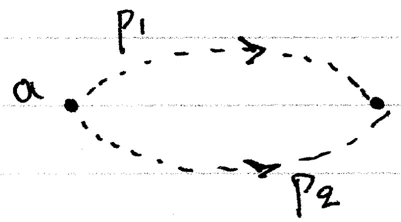
$|P(G, a \to b)| = 0 \Rightarrow G$ not connected

$\Rightarrow G$ not a tree.

which is a contradiction.

Case 2 : Assume that $|P(G, a \to b)| \geq 2$.

Choose paths $p_1, p_2 \in P(G, a \to b)$ such that $p_1 \neq p_2$.



Then p_1, p_2 together form a cycle and therefore

G cyclic $\Rightarrow G$ not a tree

which is again a contradiction.

It follows that $|P(G, a \to b)| = 1$

We have thus shown that

$\forall a, b \in V(G) : (a \neq b \Rightarrow |P(G, a \to b)| = 1)$

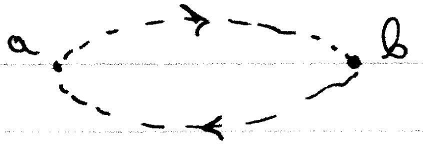
$(\Leftarrow)$: Assume that

$\forall a, b \in V(G) : (a \neq b \Rightarrow |P(G, a \to b)| = 1)$

It immediately follows that G connected. (1)

To show that G is acyclic, assume that G is cyclic.

Let p be a cycle of G and let $a, b \in V(G)$ be two vertices on the cycle p. Then, there are at least two paths that connect a to b, therefore



$|P(G, a \to b)| \geq 2$

This contradicts the hypothesis. It follows that G acyclic (2)

From Eq.(1) and Eq.(2): G is a tree.

③ → A tree always disconnects to two components upon removing any one edge

Thm: Let G be a graph. Then

$$G \text{ tree} \Rightarrow \forall e \in E(G): \omega(G-\{e\}) = 2$$

Proof

Assume that G is a tree. Let $e \in E(G)$ be given. Then

G tree $\Rightarrow$ G connected $\Rightarrow$ $\omega(G) = 1$.

We know that for any graph:

$$\omega(G) \leq \omega(G-\{e\}) \leq \omega(G)+1 \Rightarrow$$

$$\Rightarrow 1 \leq \omega(G-\{e\}) \leq 2 \Rightarrow$$

$$\Rightarrow \omega(G-\{e\}) = 1 \vee \omega(G-\{e\}) = 2 \qquad (1)$$

Let $a, b \in V(G)$ such that $\psi_G(e) = \{a, b\}$.

If $\omega(G-\{e\}) = 1 \Rightarrow G-\{e\}$ connected $\Rightarrow$

$$\Rightarrow \exists p \in P(G-\{e\}): s(p) = a \wedge t(p) = b.$$

The path p does not use the edge e. However, the edge e defines a second path $p' = (a, e, b)$.

It follows that $|P(G, a \to b)| \geq 2 \Rightarrow G$ not a tree which is a contradiction.

We conclude that $\omega(G-\{e\}) \neq 1 \qquad (2)$

From Eq.(1) and Eq.(2): $\omega(G-\{e\}) = 2$.

We have thus shown that

$\forall e \in E(G): \omega(G-\{e\}) = 2$. □

④ → Relation between number of edges and vertices

Thm : Let G be a graph. Then
$$G \text{ tree} \Leftrightarrow \begin{cases} |E(G)| = |V(G)| - 1 \\ G \text{ connected} \end{cases}$$

Proof : $(\Rightarrow)$

Assume that G is a tree. Then, by definition

$$G \text{ tree} \Rightarrow G \text{ acyclic} \wedge G \text{ connected}$$
$$\rightarrow G \text{ connected} \qquad (1)$$

To show $|E(G)| = |V(G)| - 1$ we use induction on $|V(G)|$

▸ For $|V(G)| = 1$, the only tree that can be defined is:

G :

or equivalently:
$$\begin{cases} V(G) = \{a\} \\ E(G) = \emptyset \\ \psi_G = \emptyset \end{cases}$$
Then $|E(G)| = |\emptyset| = 0 = 1 - 1 = |V(G)| - 1$

▸ For $|V(G)| \leq n$, we assume that the statement has been shown for all possible trees.

▸ Consider any tree G with $|V(G)| = n + 1$.

Choose any edge $e \in E(G)$. From property ③ we have $w(G - \{e\}) = 2$. Let G_1, G_2 be the connected components of $G - \{e\}$ such that $G - \{e\} = G_1 \cup G_2$. Note that

G_1, G_2 cannot be cyclic because otherwise G would not be a tree. It follows that

G_1, G_2 trees $\Rightarrow \begin{cases} |E(G_1)| = |V(G_1)| - 1 \\ |E(G_2)| = |V(G_2)| - 1 \end{cases}$, via the induction hypothesis.

We also note that G has the edges of G_1, G_2, and the edge e., therefore

$$\begin{aligned}|E(G)| &= |E(G_1)| + |E(G_2)| + 1 \\ &= [|V(G_1)| - 1] + [V(G_2) - 1] + 1 \\ &= |V(G_1)| + |V(G_2)| - 1 \\ &= |V(G_1 \cup G_2)| - 1 = |V(G - \{e\})| - 1 \\ &= |V(G)| - 1.\end{aligned}$$

$(\Leftarrow)$: Assume that

$$\begin{cases} |E(G)| = |V(G)| - 1 \\ G \text{ connected} \end{cases}$$

To show that G is a tree, we assume that G is not a tree, to arrive a contradiction. Remove as many edges from G as necessary to eliminate all cycles in G. Let $E_0 \subseteq E(G)$ be the set of all the edges removed. Then, using the $(\Rightarrow)$ result:

$$G - E_0 \text{ tree} \Rightarrow |E(G - E_0)| = |V(G - E_0)| - 1 \Rightarrow$$

$$\begin{aligned}\Rightarrow |E(G)| &= |E(G - E_0)| + |E_0| = \\ &= |V(G - E_0)| - 1 + |E_0| = \\ &= [|V(G)| - 1] + |E_0| = |E(G)| + |E_0| \Rightarrow\end{aligned}$$

$$\Rightarrow |E_0| = 0.$$

This contradicts the assumption that G is not a tree

It follows that G is a tree

⑤ → Trees and connectivity

Thm: Let G be a ~~tree~~ graph. Then
$$G \text{ tree} \Longrightarrow \kappa(G) = \lambda(G) = \delta(G) = 1$$

Proof

Assume that G is a tree. Then,

$|E(G)| = |V(G)| - 1 \Longrightarrow$

$$\Rightarrow \delta(G) \leq \frac{2|E(G)|}{|V(G)|} = \frac{2(|V(G)|-1)}{|V(G)|} < 2$$

$\Rightarrow \delta(G) = 0 \lor \delta(G) = 1.$

We also note that

G tree $\Rightarrow G$ acyclic $\land$ G connected

$\Rightarrow G$ connected

$\Rightarrow \kappa(G) \geq 1 \land \lambda(G) \geq 1 \land \delta(G) \geq 1$

It follows that

$\delta(G) = 1$

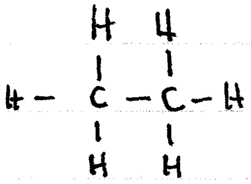
$1 \leq \kappa(G) \leq \delta(G) = 1 \Rightarrow \kappa(G) = 1$

$1 \leq \lambda(G) \leq \delta(G) = 1 \Rightarrow \lambda(G) = 1.$ $\square$

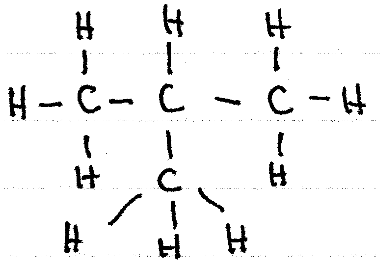
EXAMPLE / APPLICATION

A saturated hydrocarbon is a molecule C_aH_b in which
a) Every C atom has 4 simple bonds
b) Every H atom has 1 simple bond
c) No sequence of bonds forms a cycle.
Show that (a),(b),(c) imply that $b = 2a + 2$

↦ examples



C_2H_6



C_4H_{10}

Solution
Let G be the graph representing the molecule C_aH_b where the C, H atoms are vertices and the bonds are edges. Let V_1 be the vertices corresponding to C atoms and let V_2 be the vertices corresponding to H atoms. Since C atoms have 4 bonds and H atoms have 1 bond:

$$\begin{cases} \forall u \in V_1 : d(u) = 4 \\ \forall u \in V_2 : d(u) = 1 \end{cases}$$

We also note that $|V_1| = a$ and $|V_2| = b$. It follows that
$|V(G)| = |V_1| + |V_2| = a + b$

and

$$|E(G)| = \frac{1}{2}\sum_{u \in V(G)} d(u) = \frac{1}{2}\sum_{u \in V_1} d(u) + \frac{1}{2}\sum_{u \in V_2} d(u) =$$

$$= \frac{4|V_1|}{2} + \frac{|V_2|}{2} = \frac{4a+b}{2}$$

Since no sequence of bonds forms a cycle,

G is a tree $\Rightarrow |E(G)| = |V(G)| - 1 \Rightarrow \dfrac{4a+b}{2} = a + b - 1$

$$\Rightarrow 4a + b = 2(a+b) - 2$$

$$\Rightarrow 4a + b = 2a + 2b - 2$$

$$\Rightarrow 4a + b - 2a - 2b + 2 = 0$$

$$\Rightarrow 2a - b + 2 = 0 \Rightarrow \underline{b = 2a + 2.}$$

EXAMPLE / APPLICATION

Consider an unsaturated hydrocarbon molecule C_aH_b such that

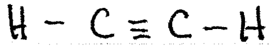
a) Every C atom has 4 simple bonds, except there also exists one triple bond between C atoms.

b) Every H atom has 1 simple bond

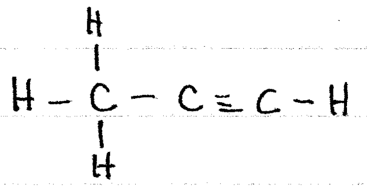
c) No sequence of bonds forms a cycle.

Then, show that $b = 2a - 2$.

↦ examples



C_2H_2



C_3H_4

Solution

Let G be a graph representing the molecule C_aH_b. Let V_1 be the vertices corresponding to C_1 atoms with 4 simple bonds. Let V_2 be the vertices corresponding to G atoms with one triple bond and one simple bond. Let V_3 be the vertices corresponding to H atoms with one simple bond. It follows that

$$\begin{cases} |V_1| + |V_2| = a \\ |V_3| = b \end{cases}$$

and

$$\begin{cases} \forall u \in V_1 : d(u) = 4 \\ \forall u \in V_2 : d(u) = 2 \\ \forall u \in V_3 : d(u) = 1 \end{cases}$$

Also, since there is only one triple bond, involving two C atoms, it follows that $|V_2| = 2$.

Then:

$$|V(G)| = |V_1| + |V_2| + |V_3| = (|V_1| + |V_2|) + |V_3| = a + b$$

and

$$|V_1| + |V_2| = a \Rightarrow |V_1| = a - |V_2| = a - 2$$

and

$$|E(G)| = \frac{1}{2} \sum_{u \in V(G)} d(u) =$$

$$= \frac{1}{2} \sum_{u \in V_1} d(u) + \frac{1}{2} \sum_{u \in V_2} d(u) + \frac{1}{2} \sum_{u \in V_3} d(u)$$

$$= \frac{4|V_1|}{2} + \frac{2|V_2|}{2} + \frac{1|V_3|}{2} =$$

$$= \frac{4(a-2) + 2 \cdot 2 + b}{2} = \frac{4a - 8 + 4 + b}{2} =$$

$$= \frac{4a + b - 4}{2}$$

Since no sequence of bonds forms a cycle, G is a tree $\Rightarrow |E(G)| = |V(G)| - 1 \Rightarrow$

$$\Rightarrow \frac{4a + b - 4}{2} = a + b - 1 \Rightarrow$$

$$\Rightarrow 4a+b-4 = 2a+2b-2 \Rightarrow$$
$$\Rightarrow 4a+b-4-2a-2b+2=0$$
$$\Rightarrow 2a-b-2=0 \Rightarrow \underline{b=2a-2}$$

EXAMPLE

a) Show that
K_a is a tree $\Leftrightarrow$ $a=1 \vee a=2$

↦ Method: For problems of this time, it is possible to construct a direct argument using $\Leftrightarrow$ (if and only if) at every step, via the following proposition:

G tree $\Leftrightarrow$ G connected $\wedge$ $|E(G)| = |V(G)|-1$

Usually, the condition that G is connected can be removed if it is unconditionally true.

Solution

K_a has "a" vertices, and every vertex connects with one edge with the other $a-1$ vertices. It follows that:

$|V(K_a)| = a$ and

$\forall u \in V(K_a): d(u) = a-1$

and therefore:

$$|E(K_a)| = \frac{1}{2}\sum_{u\in V(K_a)} d(u) = \frac{1}{2}(a-1)|V(K_a)| = \frac{a(a-1)}{2}$$

It follows that

$$|E(K_a)| - |V(K_a)| + 1 = \frac{a(a-1)}{2} - a + 1 = \frac{1}{2}\left[a^2 - a - 2a + 2\right] =$$

$$= (1/2)(a^2 - 3a + 2) = (1/2)(a-1)(a-2)$$

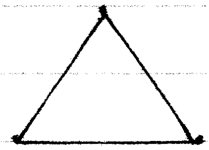
and therefore:

$$\begin{aligned}
\underline{K_a \text{ free}} &\iff K_a \text{ connected} \wedge |E(K_a)| = |V(K_a)| - 1 \\
&\iff |E(K_a)| - |V(K_a)| + 1 = 0 \\
&\iff (1/2)(a-1)(a-2) = 0 \\
&\iff a - 1 = 0 \vee a - 2 = 0 \\
&\iff \underline{a = 1 \vee a = 2}
\end{aligned}$$

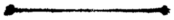
$\mapsto$ Note that:

K_1:

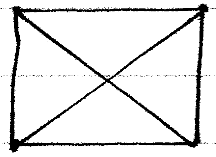
K_3:



K_2:



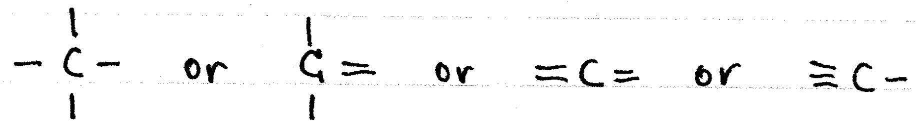
K_4:



EXERCISES

(49) Consider a generalised hydro carbon molecule C_aH_b that has t triple bonds and d double bonds. Show that: $b = 2a - 2(t+d-1)$.

↳ Hint / Outline : Possible configurations for the bonds of C atoms are:



and for H atoms: $H-$

Consequently, we define the following vertex sets:

V_1 = all C atoms with 4 simple bonds

V_2 = all C atoms with 1 simple and 1 triple bond.

V_3 = all C atoms with 2 double bonds

V_4 = all C atoms with 1 double and 2 simple bonds.

V_5 = all H atoms with 1 simple bond.

Then we note that $2|V_3| + |V_4| = 2d$ and $|V_2| = 2t$ (explain why) and show that $|V(G)| = a+b$ and $|E(G)| = \frac{1}{2}\left[2a + b + 2|V_1| + |V_4|\right]$

Then we show that $2|V_1| + |V_4| = 2a - 2t - 2d$ and the rest is up to you.

(50) Let G be a tree. Use the fact that $\delta(G)=1$ and $\kappa(G)=1$ to show that

a) G is not Eulerian

b) G is not Hamiltonian

c) Can we establish (a) and (b) via an alternate argument directly from the tree definition?

(51) A <u>forest</u> is a graph whose components $G_1, G_2, \ldots, G_n$ are trees. Show that if G is a forest, then

$$|E(G)| = |V(G)| - \omega(G)$$

(Hint: for all a, we note that $|E(G_a)| = |V(G_a)| - 1$).

(52) Let $K_{a,b}$ be the complete bipartite graph. Show that

$K_{a,b}$ is a tree $\Longleftrightarrow$ $a=1$ or $b=1$

(53) Show that the path graph P_a is a tree for all $a \geqslant 2$.

(54) Let C_a be the cycle graph. Show that for $a>2$, C_a is not a tree.

↳ Use the statement

G tree $\Longleftrightarrow (|E(G)| = |V(G)| - 1 \wedge G$ connected$)$.

(55) Let G be a regular graph with at least three vertices. Show that G cannot be a tree.

↳ Hint/Outline: We argue by contradiction. To show that G is not a tree, assume that G is a tree. Then show that if ρ is the common degree of all vertices, then ρ satisfies:

$$\rho = 2\,\frac{|V(G)|-1}{|V(G)|}$$

Next show that $|V(G)| \geqslant 3$ implies $1 < \rho < 2$ which is a contradiction since ρ has to be an integer. It will help to write $\rho = 2f(|V(G)|)$ with $f(x) = \dfrac{x-1}{x}$ and use calculus to study the graph of f in the interval $[3, +\infty)$.

▼ The minimum spanning tree problem

- Let G be a graph. We say that a tree T is a <u>spanning tree</u> of G if and only if

  (a) T is a tree
  (b) T is a subgraph of G
  (c) All the vertices of G are also vertices of T, and vice versa.

$$\boxed{T \text{ spanning tree of } G \iff \begin{cases} T \subseteq G \\ T \text{ is a tree} \\ V(T) = V(G) \end{cases}}$$

- The set of all spanning trees of G is denoted as

$$\boxed{\tau(G) = \{T \subseteq G \mid T \text{ spanning tree of } G\}}$$

- <u>Thm</u> : (Cayley) The complete graph K_n has n^{n-2} spanning trees.

$$\boxed{|\tau(K_n)| = n^{n-2}}$$

- <u>The problem</u> : Let $f: E(G) \to \mathbb{R}$ be a weight function that maps every edge $e \in E(G)$ to a number $f(e) \in \mathbb{R}$.

If $T \in \tau(G)$ is a spanning tree of G then the weight associated with T is given by

$$\boxed{w(T) = \sum_{e \in E(T)} f(e)}$$

A tree T_0 is a minimum spanning tree if and only if it minimizes $f(T)$.

$$\boxed{T_0 \text{ minimum spanning tree of } G \iff \forall T \in \tau(G) : f(T_0) \leq f(T).}$$

- A graph always has at least one minimum spanning tree and it is not necessarily unique.

- <u>Kruskal's Algorithm</u>

1) Choose $e_1 \in E(G)$ that minimizes $f(e_1)$
2) Assume we have chosen $e_1, e_2, \dots, e_k$.
   Choose $e_{k+1} \in E(G) - \{e_1, e_2, \dots, e_k\}$ such that

(a) $f(e_{k+1})$ is minimum

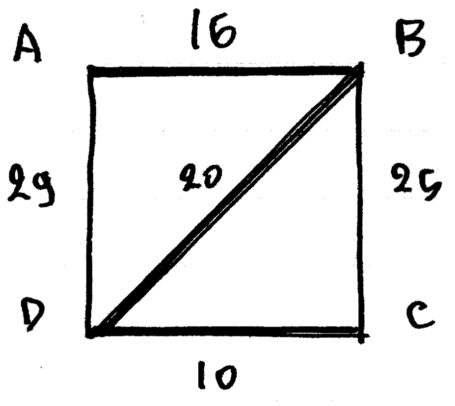
(b) The induced graph $G[\{e_1, e_2, \dots, e_{k+1}\}]$ is acyclic.

3) Repeat 2 until e_{k+1} cannot be found.

Upon completion, we have the edges $e_1, e_2, \dots, e_n$ and the minimum spanning tree is:

$$T_0 = G[\{e_1, e_2, \dots, e_n\}]$$

<u>example</u>

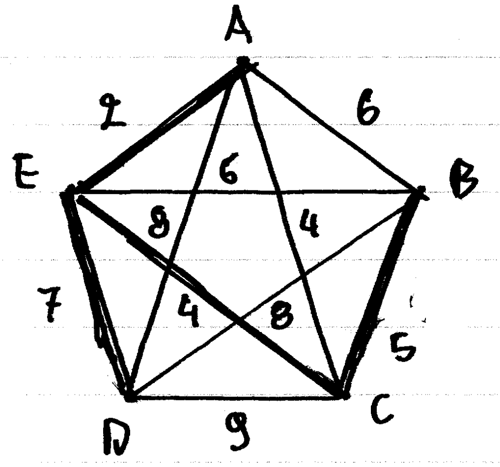


Add DC
Add AB
Add BP

Minimum spanning tree.

$$T_0 = G[\{AB, BP, DC\}]$$

EXAMPLE



Add AE
From AC, CE choose CE
Reject AC (cycle AECA)
Add BC
Reject AB (cycle AECBA)
Reject BE (cycle ECBE)
Add ED
We now have a spanning tree

Thus : $T_0 = G[\{AE, ED, EC, BC\}]$

EXERCISES

(56) Show that if G connected then

$$|E(G)| \geqslant |V(G)| - 1$$

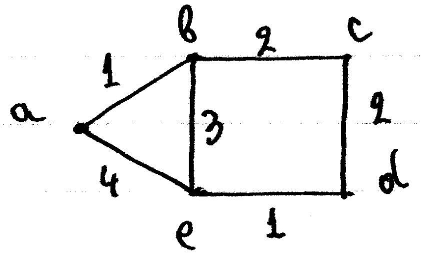
(Hint: Consider the spanning tree of G)

(57) How many spanning trees does the following graphs have?

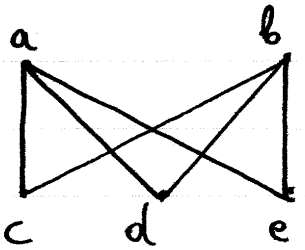
a) K_3 b) K_4 c) K_5

(58) Find the minimum spanning tree for the following graphs:

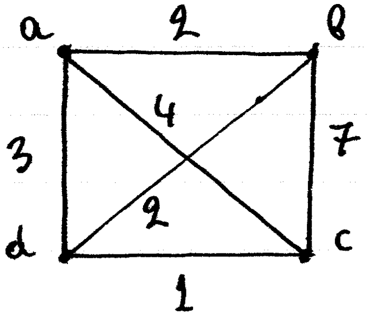
a)



c)



b)



with

$ac = 1$, $ad = 3$, $ae = 2$
$bc = 4$, $bd = 5$, $be = 3$

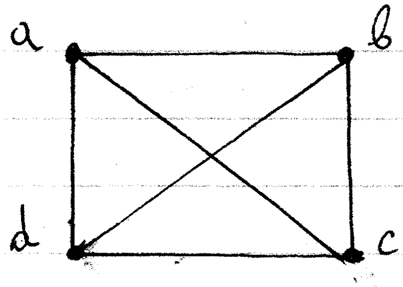
(59) Show that $|\tau(K_{2,n})| = n2^{n-1}$.
(Hint: Try $K_{2,4}$ first as an example then generalize)

▼ Planar graphs

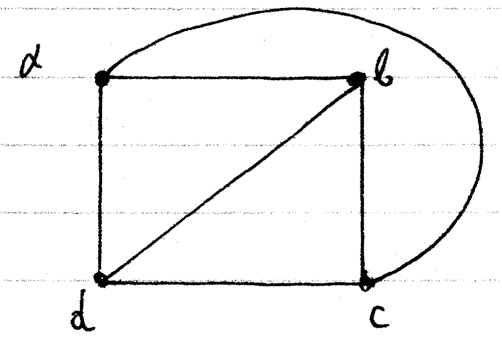
Let G be a graph. We say that G is <u>planar</u> if and only if it can be embedded on a two-dimensional plane so that no two edges intersect, except at the vertices. A graph whose definition does not seem to be planar could still be planar, using a different embedding. The definition of "planar" requires the existence of at least one such embedding.

EXAMPLE

K_4 is planar. Consider an embedding which is not planar and a different embedding which is planar.



non-planar embedding of K_4

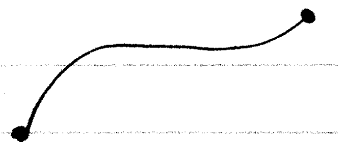


planar embedding of K_4

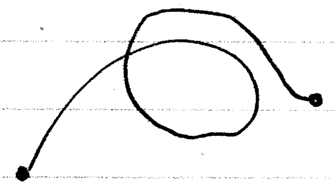
● → Rigorous definition

A rigorous definition of planar graphs is based on the concept of a <u>simple curve</u>. Intuitively, a simple curve is defined as a curve that does not cross itself. We also need the curve to be continuous, with no interruptions.

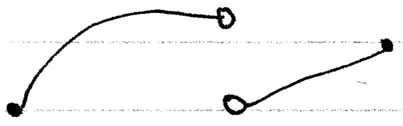
EXAMPLE



"simple curve"



not simple curve



not simple due to lack of continuity.

Def (Simple curves)

(a) A curve (c) defined as

$$(c) : (x,y) = f(t) \quad , \quad t \in [a,b]$$

is the set of points given by:

$$C(c) = \{ f(t) \mid t \in [a,b] \} =$$
$$= \{ (x,y) \in \mathbb{R}^2 \mid \exists t \in [a,b] : (x,y) = f(t) \}$$

(b) We say that

$$(c) \text{ simple curve} \Leftrightarrow \begin{cases} f \text{ continuous on } [\alpha, b] \\ f \text{ one-to-one} \end{cases}$$

Remarks

- We denote $C_2[a,b]$ as the set of all mappings $f:[a,b]\to\mathbb{R}^2$ that define a simple curve.
- Note that given two such mappings $f,g\in C_2[0,1]$, the claim that the corresponding curves do not intersect, except possibly at the endpoints can be written concisely as:

$$f((0,1))\cap g((0,1))=\emptyset$$

Using open intervals is needed in order to ignore the endpoints of the two curves.

Based on the above, we define planar graphs as follows:

Def : Let G be a graph. We say that G is planar if and only if there exist

(a) a mapping $f:V(G)\to\mathbb{R}^2$ of vertices to points on the plane $\mathbb{R}^2$

(b) a mapping $g:E(G)\longrightarrow C_2[0,1]$ of edges onto mappings that represent simple curves.

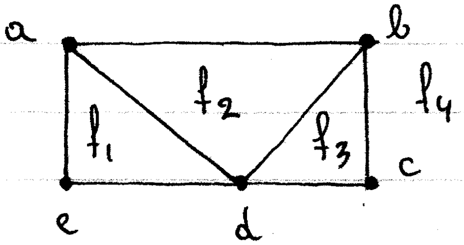
such that the following condition is satisfied:

$$\begin{cases}\forall e\in E(G):\ [g(e)](\{0,1\})=f(\psi_G(e))\\ \forall e_1,e_2\in E(G):\ [g(e_1)]((0,1))\cap[g(e_2)]((0,1))=\emptyset\end{cases}$$

Faces of planar graphs

- A planar graph partitions the plane into regions. We call these region faces and the set of all faces of the graph G is denoted F(G). This includes the infinite face which represents the area outside the overall perimeter of the graph G.

EXAMPLE



$F(G) = \{f_1, f_2, f_3, f_4\}$
with f_4 the infinite face.

- For every edge $e \in E(G)$, on either side of the edge e there are one or two faces. We say that those faces are incident upon the edge e, and we define on incidence mapping

$$f_G : E(G) \to \mathcal{P}_1(F(G)) \cup \mathcal{P}_2(F(G))$$

such that

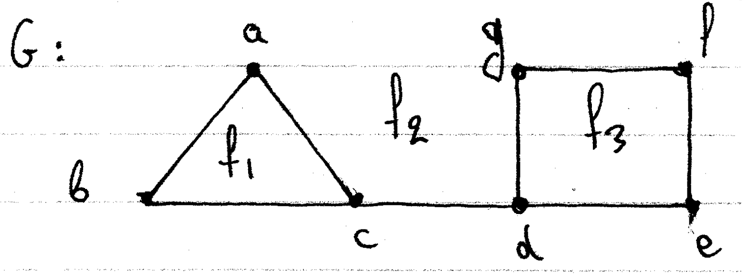
$$\forall e \in E(G) : f_G(e) = \{ f_0 \in F(G) \mid f_0 \text{ incident to } e \}$$

Def: Let G be a planar graph and let $f_1, f_2 \in F(G)$ be two faces of G. We say that

f_1, f_2 adjacent $\iff \exists e \in E(G) : f_G(e) = \{f_1, f_2\}$

EXAMPLE

Consider the graph



Then:

$$\begin{cases} f_G(ab) = f_G(bc) = f_G(ac) = \{f_1, f_2\} \\ f_G(cd) = \{f_2\} \\ f_G(de) = f_G(ef) = f_G(fg) = f_G(gd) = \{f_2, f_3\}. \end{cases}$$

- Recall that given a connected graph G and an edge $e \in E(G)$, we say that
  $$e \text{ bridge} \iff \omega(G - \{e\}) > \omega(G)$$
  It can be shown that:

Prop: Let G be a planar graph and let $e \in E(G)$ be an edge. Then:
$$e \text{ bridge} \iff |f_G(e)| = 1$$

Dual graph and face degree

Def: Let G be a planar graph. We define the dual graph G^* such that:

$$\begin{cases} V(G^*) = F(G) \\ E(G^*) = E(G) \\ \forall e \in E(G^*) : \ \psi_{G^*}(e) = f_G(e) \end{cases}$$

Informally, the vertices of G^* are the faces of G. G^* and G have the same edges. In G^* and edge connects faces of G if in G the two faces are incident to that edge. The definition of G^* allows us to define the degree of faces as follows:

Def: Let G be a planar graph with dual graph G^*. Let $f \in F(G)$ be a face of G. We define the degree $d_G(f)$ such that:

$$d_G(f) = d_{G^*}(f)$$

Note that G^* sees f as a vertex and thus $d_{G^*}(f)$ is the vertex degree of f ~~not~~ with respect to G^*, which was previously defined. An immediate consequence is a handshaking lemma for faces.

Lemma: Let G be a planar graph. Then,

$$\sum_{f \in F(G)} d_G(f) = 2|E(G)|$$

Proof

Let G^* be the dual graph of G. Then:

$$\sum_{f \in F(G)} d_G(f) = \sum_{f \in V(G^*)} d_{G^*}(f) \quad [\text{def. of face degree}]$$
$$= 2|E(G^*)| \quad [\text{handshaking lemma on } G^*]$$
$$= 2|E(G)| \quad [\text{definition of } G^*]$$

which prores the lemma.

Def: Let G be a planar graph. We say that

a) G face-regular $\Leftrightarrow \exists a \in \mathbb{N} : \forall f \in F(G) : d_G(f) = a$

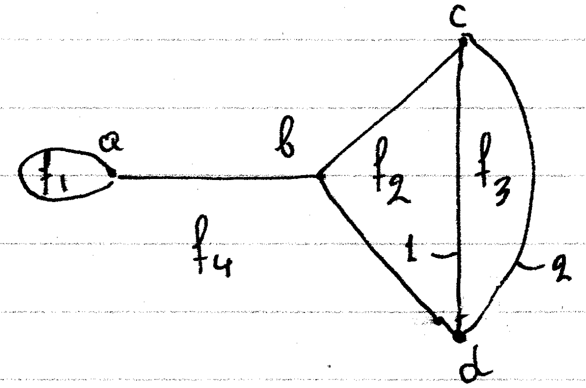
b) G face-regular with regularity $a \in \mathbb{N} \Leftrightarrow$
$\Leftrightarrow \forall f \in F(G) : d_G(f) = a$

An immediate consequence of the handshaking lemma for faces is that

Prop: Let G be a planar graph. Then
G face-regular with face-regularity $\alpha \in \mathbb{N} \Rightarrow$
$\Rightarrow a|F(G)| = 2|E(G)|$

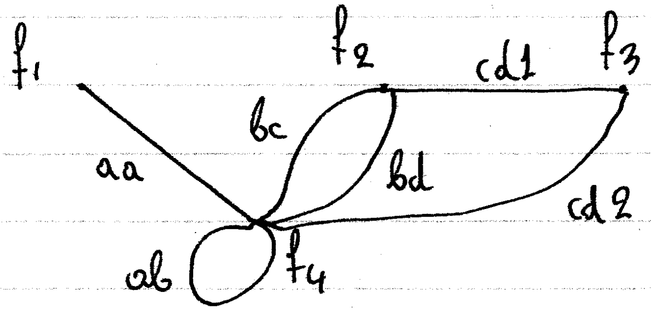
EXAMPLE

For G:



the dual graph is:

G^*:



- Note that a loop in G becomes a bridge in G^*.
- A bridge in G becomes a loop in G^*
- $d_G(f_1) = |\{aa\}| = 1$
  $d_G(f_2) = |\{bc, bd, cd1\}| = 3$
  $d_G(f_3) = |\{cd1, cd2\}| = 2$
  $d_G(f_4) = |\{aa, ab, bc, bd, cd2\}| + |\{ab\}| =$
  $= 5 + 1 = 6.$

Face Boundary and face degree

- Let G be a planar graph. Intuitively, a walk $w \in W(G)$ is the boundary of a face $f \in F(G)$ if and only if it is a closed walk that uses every edge that is not a bridge once, and uses every edge that is indeed a bridge twice, and includes all edges that are incident to the face f.
- The set of all such walks shall be denoted as $b(f)$.

Writing this definition rigorously is difficult, but it can be effected by introducing some edge-counting notation:

notation: Let G be a graph, let $w \in W(G)$ be a walk on G, and let $e \in E(G)$ be an arbitrary edge. The number of times the edge e is used on the walk w is denoted as $|e|_w$ with

$$\forall e \in E(G): |e|_w = |\{k \in [\ell(w)] \mid e_k(w) = e\}|$$

Def: Let G be a planar graph, let $f \in F(G)$ be a face, and let $w \in W(G)$ be a walk. We say that

$$w \in b(f) \iff \begin{cases} w \text{ closed walk} \\ \forall e \in E(G): (f \in f_G(e) \iff e \in E(w)) \\ \forall e \in E(w): |e|_w = \begin{cases} 1 & \text{, if } e \text{ is not bridge} \\ 2 & \text{, if } e \text{ is bridge} \end{cases} \end{cases}$$

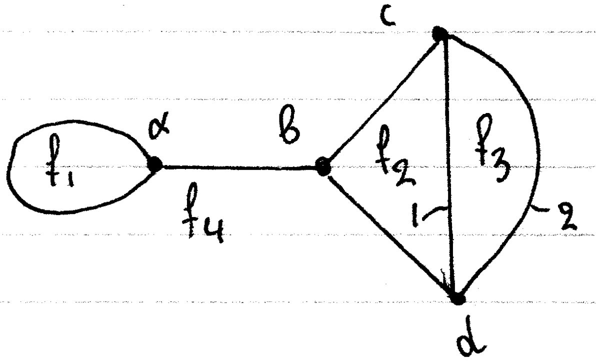
The main result is that the length of all walks $W \in \mathcal{B}(f)$ is equal to $d_G(f)$

Prop: Let G be a planar graph, and let $f \in F(G)$ be a face of G. Then
$\forall w \in \mathcal{B}(f): \ l(w) = d_G(f)$

EXAMPLE

Consider the previous example

G:



$(a, \underline{aa}, a) \in \mathcal{B}(f_1) \Rightarrow d_G(f_1) = 1$

$(b, \underline{bc}, c, \underline{cd1}, d, \underline{db}, b) \in \mathcal{B}(f_2) \Rightarrow d_G(f_2) = 3$

$(c, \underline{cd2}, d, \underline{cd1}, c) \in \mathcal{B}(f_3) \Rightarrow d_G(f_3) = 2$

$(a, \underline{aa}, a, \underline{ab}, b, \underline{bc}, c, \underline{cd2}, d, \underline{bd}, b, \underline{ab}, a) \in \mathcal{B}(f_4)$
$\Rightarrow d_G(f_4) = 6$

● → Properties of graphs

① → Trees are planar

Thm: Let G be a graph. Then

$$G \text{ tree} \Rightarrow \begin{cases} G \text{ planar} \\ |F(G)| = 1 \end{cases}$$

To show that a tree is planar, we need to define a general embedding for all trees onto $\mathbb{R}^2$. This can be done by choosing a vertex and exploiting the length of the unique path to all other vertices. To show that $|F(G)| = 1$, we note that G has the infinite face, so $|F(G)| \geqslant 1$. To show that $|F(G)| < 2$ we note that if G has additional faces, those faces are enclosed by cycles, and if G has cycles then it is not a tree. This proof sketch can be made more precise, but doing so is a lot of work.

② → Euler's formula

Thm: Let G be a graph. Then

$$\begin{cases} G \text{ planar} \\ G \text{ connected} \end{cases} \Rightarrow |V(G)| - |E(G)| + |F(G)| = 2$$

Proof

We use the Kruskal algorithm to remove edges that are part of cycles in G until all cycles are eliminated and we obtain a spanning tree T. Let E_0 be the set of all edges removed. Since every edge $e \in E_0$ was part of some cycle, removing each $e \in E_0$ corresponded to merging twofaces, thus reducing the number of faces by 1. It follows that $|F(G)| - |E_0| = |F(G - E_0)| = |F(T)| = 1$, since T is a tree. We also note that since

$$T \text{ spanning tree of } G \Rightarrow \begin{cases} |V(T)| = |V(G)| \\ |E(T)| = |V(T)| - 1 \end{cases}$$

and since

$$T = G - E_0 \Rightarrow |E(G)| = |E(T)| + |E_0|$$

From the above, we have:

$$\begin{aligned} |V(G)| - |E(G)| + |F(G)| &= |V(T)| - [|E(T)| + |E_0|] + [1 + |E_0|] \\ &= |V(T)| - |E(T)| - |E_0| + 1 + |E_0| = \\ &= |V(T)| - |E(T)| + 1 = \\ &= |V(T)| - [|V(T)| - 1] + 1 = \\ &= |V(T)| - |V(T)| + 1 + 1 = \\ &= 1 + 1 = 2. \end{aligned}$$

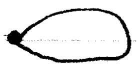
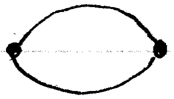
③ → Planarity and girth

Def: Let G be a graph. We define the girth $g(G)$ of the graph G as:

$$g(G) = \min\{n \in \mathbb{N}^* \mid \exists H \in \mathcal{P}(G) : H \cong C_n\}$$

Remarks

a) The girth $g(G)$ is equal to the length of the shortest cycle C_n contained in the graph G. Recall that:

C_1 :  and C_2: 

b) If G has a loop then $g(G) = 1$
If G has a multiple edge and no loops, then $g(G) = 2$.
If G is simple (no loops and no multiple edges) then $g(G) \geq 3$.

c) For any face $f \in F(G)$ the length of a boundary $W \in \mathcal{B}(f)$ cannot be shorter than $g(G)$. It follows that:

$$\boxed{G \text{ planar} \Rightarrow \forall f \in F(G) : d_G(f) \geq g(G)}$$

Our main result is the following theorem:

Thm: Let G be a graph. Then:

$$\left.\begin{cases} G \text{ planar} \\ G \text{ connected} \\ G \text{ simple}\end{cases}\right. \Rightarrow |E(G)| \leq \frac{g(G)[|V(G)|-2]}{g(G)-2}$$

Remarks

a) In practice, we use the contrapositive statement to show that a graph is NOT planar:

$$\left.\begin{array}{l} G \text{ connected} \wedge G \text{ simple} \\ |E(G)| > \dfrac{g(G)[|V(G)-2]}{g(G)-2} \end{array}\right\} \Rightarrow G \text{ not planar}$$

b) It is worth noting that

$$\left.\begin{cases} G \text{ connected} \\ G \text{ simple}\end{cases}\right. \Rightarrow |V(G)| \geq 3$$

which ensures the numerator of the RHS of the inequality is positive. Likewise,

$$G \text{ simple} \Rightarrow g(G) \geq 3$$

so the denominator will be positive too.

Proof of theorem

Assume that G is planar, connected, and simple. Since $\forall f \in F(G) : d_G(f) \geq g(G)$

$$\Rightarrow 2|E(G)| = \sum_{f \in F(G)} d_G(f) \geq \sum_{f \in F(G)} g(G) = g(G)|F(G)| \Rightarrow$$

$$\Rightarrow |F(G)| \leqslant \frac{2}{g(G)} |E(G)|$$

and since

$$\left\{\begin{array}{l} G \text{ planar} \\ G \text{ connected} \end{array}\right. \Rightarrow |V(G)| - |E(G)| + |F(G)| = 2 \Rightarrow$$

$$\Rightarrow |E(G)| = -2 + |V(G)| + |F(G)| \leqslant$$

$$\leqslant -2 + |V(G)| + \frac{2}{g(G)} |E(G) \Rightarrow$$

$$\Rightarrow \left[1 - \frac{2}{g(G)}\right] |E(G)| \leqslant |V(G)| - 2$$

$$\Rightarrow \frac{g(G) - 2}{g(G)} |E(G)| \leqslant |V(G)| - 2 \qquad (1)$$

and since

$$G \text{ simple} \Rightarrow g(G) \geqslant 3 \Rightarrow g(G) - 2 \geqslant 3 - 2 = 1 > 0 \qquad (2)$$

combining Eq.(1) and Eq.(2); we get:

$$|E(G)| \leqslant \frac{g(G)[|V(G)| - 2]}{g(G) - 2} \qquad \square$$

↳ Note that with $x = g(G)$, the function

$$f(x) = \frac{x}{x-2}, \quad \forall x \in [3, +\infty)$$

has derivative

$$f'(x) = \frac{d}{dx}\left[\frac{x}{x-2}\right] = \frac{(x)'(x-2) - x(x-2)'}{(x-2)^2}$$

$$= \frac{(x-2)-x}{(x-2)^2} = \frac{-2}{(x-2)^2} < 0 \quad , \quad \forall x \in [3,+\infty) \Rightarrow$$

$\Rightarrow f$ decreasing on $[3,+\infty)$

This means that the inequality in the above theorem becomes tighter as we increase $g(G)$. It also follows that:

$$\boxed{\left.\begin{cases} G \text{ planar} \\ G \text{ connected} \\ G \text{ simple} \end{cases}\right. \Rightarrow |E(G)| \leq 3(|V(G)|-2)}$$

<u>Proof</u>

Since $\begin{cases} G \text{ simple} \\ G \text{ connected} \end{cases} \Rightarrow \begin{cases} g(G) \geq 3 \\ |V(G)| \geq 3 \end{cases}$

it follows that

$$\begin{cases} G \text{ planar} \\ G \text{ connected} \\ G \text{ simple} \end{cases} \Rightarrow |E(G)| \leq \frac{g(G)[|V(G)|-2]}{g(G)-2} =$$

$$= f(g(G))\,[|V(G)|-2]$$

$$\leq f(3)\,[|V(G)-2]$$

$$= \frac{3}{3-2}\,[|V(G)|-2]$$

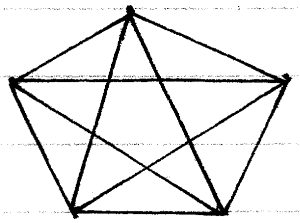
$$= 3(|V(G)|-2) \qquad \square$$

APPLICATIONS

a) Show that K_5 is not planar.

Solution

K_5:



We note that $|V(K_5)| = 5$ and $\forall u \in V(K_5): d(u) = 4$, so

$$2|E(K_5)| = \sum_{u \in V(K_5)} d(u) = \sum_{u \in V(K_5)} 4 = 4|V(K_5)| = 4 \cdot 5 = 20 \Rightarrow$$

$\Rightarrow |E(K_5)| = 10$

Also

$3(|V(K_5)| - 2) = 3(5-2) = 3 \cdot 3 = 9 < 10 = |E(K_5)| \Rightarrow$

$\Rightarrow |E(K_5)| > 3(|V(K_5)| - 2)$ (1)

and

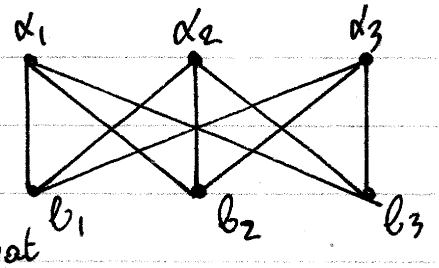
$\begin{cases} K_5 \text{ connected} \\ K_5 \text{ simple} \end{cases}$ (2)

From Eq.(1) and Eq.(2): K_5 not planar.

8) Show that $K_{3,3}$ is not planar

Solution

$K_{3,3}$:



We note that

$|V(K_{3,3})| = 3+3 = 6$

and

$\forall u \in V(K_{3,3}) : d(u) = 3$

$$\Rightarrow 2|E(K_{3,3})| = \sum_{u \in V(K_{3,3})} d(u) = \sum_{u \in V(K_{3,3})} 3 = 3|V(K_{3,3})| =$$

$$= 3 \cdot 6 = 18 \Rightarrow$$

$\Rightarrow |E(K_{3,3})| = 9.$

Since $K_{3,3}$ simple $\Rightarrow g(G) \geq 3$.

▶ We claim that $K_{3,3}$ cannot have a 3-cycle. To show this, assume that

$w = (u_1, e_1, u_2, e_2, u_3, e_3, u_1)$

is a 3-cycle. We distinguish between the following cases.

Case 1: Assume that $u_1 \in V_1$. Then,

$u_1 \in V_1 \Rightarrow u_2 \in V_2 \rightarrow u_3 \in V_1 \Rightarrow u_1 \in V_2$

which is a contradiction.

Case 2: Assume that $u_1 \in V_2$. Then,

$u_1 \in V_2 \Rightarrow u_2 \in V_1 \Rightarrow u_3 \in V_2 \Rightarrow u_1 \in V_1$

which is also a contradiction.

In both cases we have a contradiction, so it follows that $K_{3,3}$ has no 3-cycles, therefore $g(G) \geqslant 4$.

We exhibit a 4-cycle:

$w = (a_1, a_1b_1, b_1, a_2b_1, a_2, a_2b_2, b_2, a_1b_2, a_1)$

$\Rightarrow g(G) \leqslant 4$.

It follows that $g(K_{3,3}) = 4$, and therefore:

$$\left.\begin{array}{l} \dfrac{g(K_{3,3})[|V(K_{3,3})|-2]}{g(K_{3,3})-2} = \dfrac{4(6-2)}{4-2} = \dfrac{4 \cdot 4}{2} = 8 \\ |E(K_{3,3})| = 9 \end{array}\right\} \Rightarrow$$

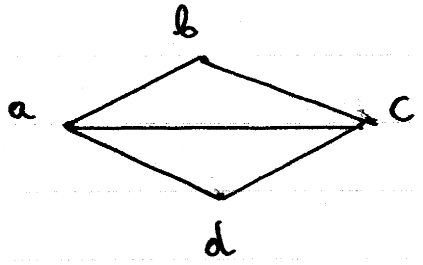
$$\Rightarrow |E(K_{3,3})| > \frac{g(K_{3,3})[|V(K_{3,3})|-2]}{g(K_{3,3})-2} \Rightarrow$$

$\Rightarrow K_{3,3}$ not planar.

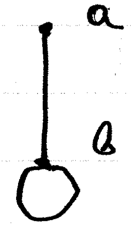
EXERCISES

(60) For the following planar graphs, identify the faces, the degree of each face and then draw the dual graph.

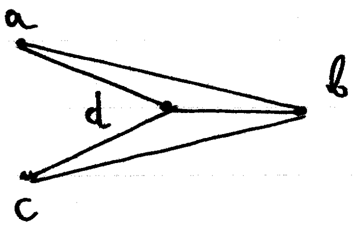
a)



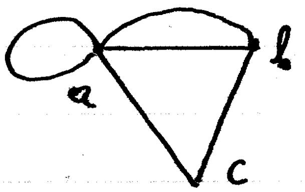
b)



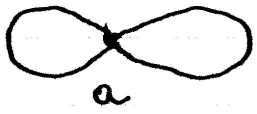
c)



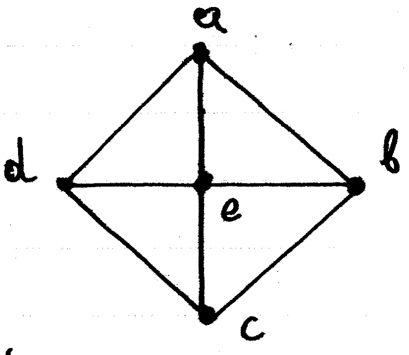
d)



e)



f)



(61) Show that if a connected planar graph is face-regular with face-regularity 4 and has 10 vertices then it must have 8 faces.

(62) Show that a face-regular planar connected graph G with face regularity r must satisfy

$$(a-2)|F(G)| = 2|V(G)|-4$$

Then show that if $|V(G)| \geqslant 3$ then $a \geqslant 3$.

(63) Consider a planar graph G which is simple, connected and regular with regularity a and face-regular with regularity b. Show that $2a+2b-ab$ divides $2ab$.
(Hint: First show that

$$(2a+2b-ab)|E(G)| = 2ab)$$

(64) Show that the following graphs are planar:

a) K_2 b) K_4
b) K_3 c) $K_{2,a}$, for $a \geqslant 1$

(65) Having shown that $K_{2,a}$ is planar, how many faces does it have, as a function of a?

(66) Show that

$a \geqslant 5 \Rightarrow K_a$ not planar
$a \geqslant 3 \Rightarrow K_{3,a}$ not planar
$a \geqslant 3$ and $b \geqslant 3 \Rightarrow K_{a,b}$ not planar.

(67) Show that a regular graph with regularity $r > 6$ which is also simple and connected can never be a planar graph.

(68) Show that if G is a planar connected graph which is face regular with face regularity a, then

a) $a-2$ divides $2|V(G)|-4$

b) $a=2 \Rightarrow |V(G)|=2$

c) $|V(G)| \geqslant 2 \Rightarrow a \geqslant 2$

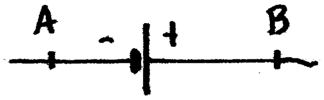
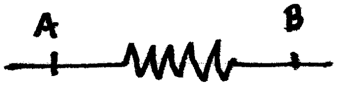
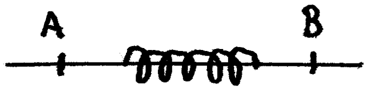
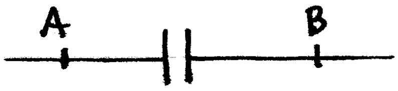
[Hint: First show that

$$(a-2)|F(G)| = 2|V(G)|-4).$$

APPLICATIONS OF LINEAR SYSTEMS

▼ DC circuits

- Every circuit component is associated with a voltage drop accross that component. The circuit components we are interested are:

Name	Notation	$V_{AB} \equiv V_A - V_B$
Generator		$V_{AB} = -E$
Resistor		$V_{AB} = IR$
Inductor		$V_{AB} = L\frac{dI}{dt}$
Capacitor		$V_{AB} = \frac{1}{C}\int_{-\infty}^{t} I d\tau$
	$\longrightarrow I$	

V_{AB} = voltage drop from A to B

E = voltage of DC generator

R = resistance

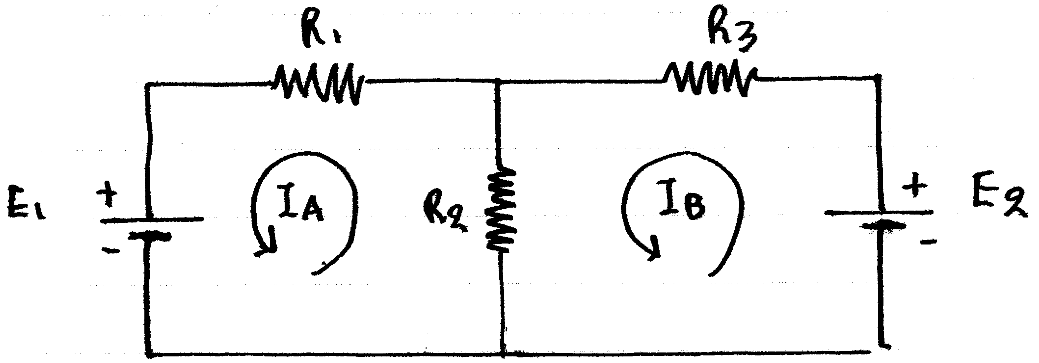
L = inductance

G = capacitance.

▶ Method: To calculate the currents around a circuit we work as follows.

- •1 Define loop currents I_A, I_B, etc. associated with each loop.
- •2 For each, the sum all of all voltage drops around the loop must be zero. Thus, for each loop, we have an equation.
- •3 Solve the system of equations to find the loop currents.
- •4 From the loop currents we may calculate the branch currents, voltage drops, etc.

example



$$\begin{array}{l} \text{Loop A:} \\ \text{Loop B:} \end{array} \begin{cases} +E_1 + I_A R_1 + (I_A - I_B) R_2 = 0 \\ -E_2 + I_B R_3 + (I_B - I_A) R_2 = 0 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} (R_1 + R_2) I_A + (-R_2) I_B = -E_1 \\ (-R_2) I_A + (R_2 + R_3) = E_2 \end{cases}$$

$$\Leftrightarrow \underbrace{\begin{bmatrix} R_1+R_2 & -R_2 \\ -R_2 & R_2+R_3 \end{bmatrix}}_{A} \begin{bmatrix} I_A \\ I_B \end{bmatrix} = \begin{bmatrix} -E_1 \\ E_2 \end{bmatrix}$$

$$\det A = \begin{vmatrix} R_1+R_2 & -R_2 \\ -R_2 & R_2+R_3 \end{vmatrix} =$$

$$= (R_1+R_2)(R_2+R_3) - (-R_2)^2 =$$

$$= R_1R_2 + R_1R_3 + R_2^2 + R_2R_3 - R_2^2 =$$

$$= R_1R_2 + R_2R_3 + R_3R_1 \neq 0$$

since $R_1 > 0$, $R_2 > 0$ and $R_3 > 0$.

Thus:

$$\begin{bmatrix} I_A \\ I_B \end{bmatrix} = \frac{1}{\det A} \begin{bmatrix} R_2+R_3 & +R_2 \\ +R_2 & R_1+R_2 \end{bmatrix} \begin{bmatrix} -E_1 \\ E_2 \end{bmatrix}$$

$$= \frac{1}{\det A} \begin{bmatrix} -E_1(R_2+R_3) + E_2R_2 \\ -E_1R_2 + E_2(R_1+R_2) \end{bmatrix}$$

so

$$I_A = \frac{-E_1(R_2+R_3) + E_2R_2}{R_1R_2 + R_2R_3 + R_3R_1}$$

$$I_B = \frac{-E_1R_2 + E_2(R_1+R_2)}{R_1R_2 + R_2R_3 + R_3R_1}$$

- Current through R_1 : I_A
  R_2 : $I_A - I_B$
  R_3 : I_B.

- Find the necessary and sufficient condition so there is no current through R_2.

Balance $\Leftrightarrow I_A - I_B = 0 \Leftrightarrow I_A = I_B \Leftrightarrow$

$$\Leftrightarrow -E_1(R_2 + R_3) + E_2R_2 = E_1R_2 + E_2(R_1 + R_2)$$

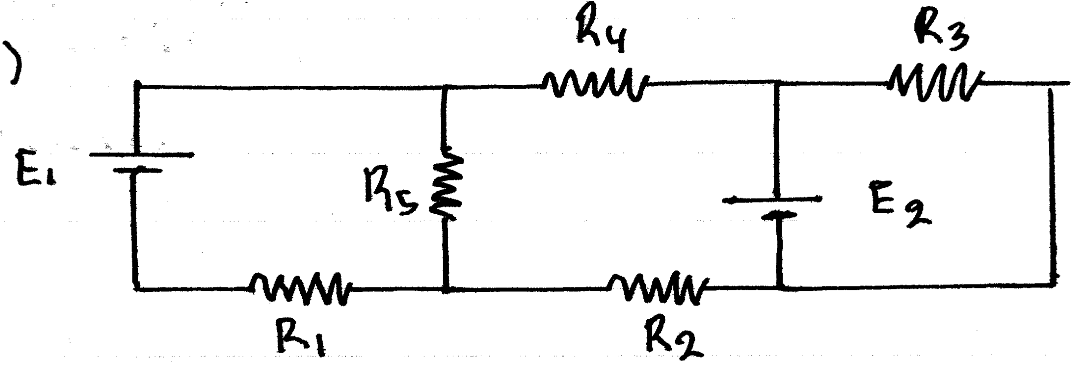
$$\Leftrightarrow \underline{E_1R_2} + E_1R_3 - \underline{\underline{E_2R_2}} = \underline{E_1R_2} - E_2R_1 - \underline{\underline{E_2R_2}}$$

$$\Leftrightarrow E_1R_3 = -E_2R_1 \Leftrightarrow \boxed{E_1 = -E_2\frac{R_1}{R_3}}$$

EXERCISES

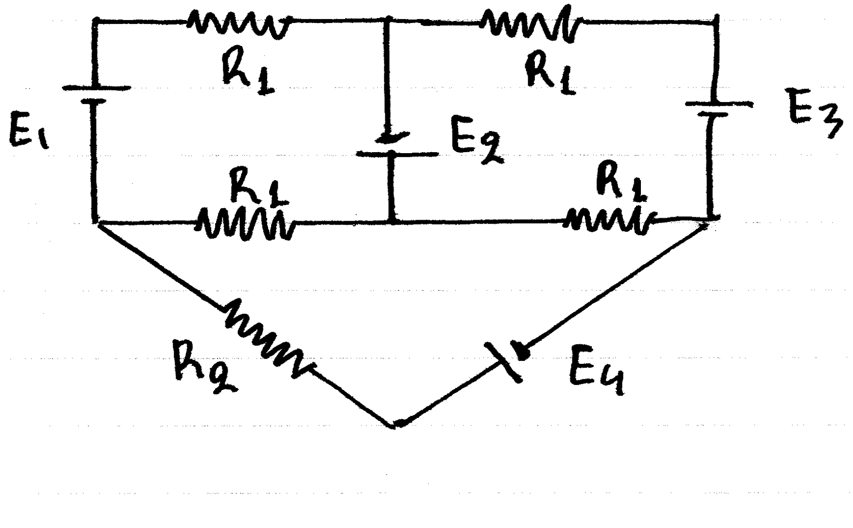
(1) Find the loop currents in the following circuits.

a)

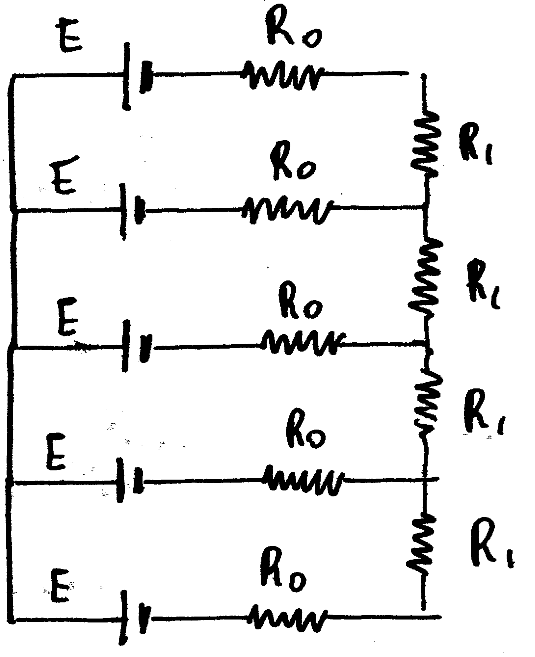


when $R_1 = R_2 = R_3 = a$
$R_4 = R_5 = b$

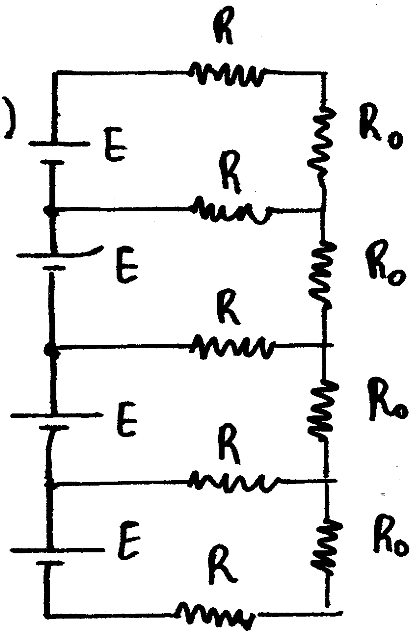
b)



c)



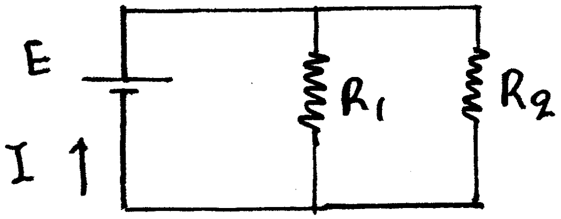
d)



② Parallel Resistors.

Show that two resistors R_1, R_2 connected in parallel to a generator are equivalent to a single resistor R with

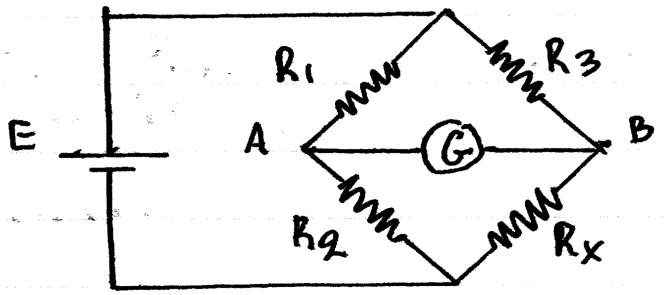
$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \qquad (1)$$



(i.e. $E = IR$ with R given by (1)).

③ Wheatstone Bridge

The Wheatstone bridge can be used to measure the unknown resistance R_X.



Ⓖ is a galvanometer measuring voltage drop V_{AB} between A and B.

Assume the Galvanometer has resistance R_g.

(a) Solve for the 3 loop currents in general.

(b) Show that when $V_{AB} = 0$ then

$$R_X = (R_2/R_1)R_3.$$

Superposition principle

- Any general DC circuit with n loops and m generators gives a linear system of equations of the form

$$\boxed{\mathcal{R}\mathcal{J} = P\mathcal{E}}$$

with $\mathcal{R}$ the resistance matrix $(\mathcal{R} \in M_{nn}(\mathbb{R}))$
P the source configuration matrix $(P \in M_{nm}(\mathbb{R}))$

and

$$\mathcal{J} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \quad \text{and} \quad \mathcal{E} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_m \end{bmatrix}$$

- If $\mathcal{R}$ has an inverse then this system has a unique solution

$$\mathcal{J} = \mathcal{R}^{-1} P \mathcal{E}$$

- <u>Thm</u> : Let $\mathcal{J}_1$ be the solution when all generators are turned off except E_1, and similarly for $\mathcal{J}_2, \mathcal{J}_3, \cdots, \mathcal{J}_n$
Then

$$\mathcal{J} = \mathcal{J}_1 + \mathcal{J}_2 + \cdots + \mathcal{J}_n$$

Proof

Define

$$\mathcal{E}_1 = \begin{bmatrix} E_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathcal{E}_2 = \begin{bmatrix} 0 \\ E_2 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \mathcal{E}_m = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ E_m \end{bmatrix}$$

Then

$$\mathcal{E} = \mathcal{E}_1 + \mathcal{E}_2 + \cdots + \mathcal{E}_m$$

For each problem we get a linear system. Altogether:

$$\begin{cases} \mathcal{R}\mathcal{J}_1 = P\mathcal{E}_1 \\ \mathcal{R}\mathcal{J}_2 = P\mathcal{E}_2 \\ \vdots \\ \mathcal{R}\mathcal{J}_m = P\mathcal{E}_m \end{cases} \Rightarrow \begin{cases} \mathcal{J}_1 = \mathcal{R}^{-1}P\mathcal{E}_1 \\ \mathcal{J}_2 = \mathcal{R}^{-1}P\mathcal{E}_2 \\ \vdots \\ \mathcal{J}_m = \mathcal{R}^{-1}P\mathcal{E}_m \end{cases}$$

It follows that

$$\mathcal{J} = \mathcal{R}^{-1}P\mathcal{E} = \mathcal{R}^{-1}P\left(\mathcal{E}_1 + \mathcal{E}_2 + \cdots + \mathcal{E}_m\right)$$

$$= \mathcal{R}^{-1}P\mathcal{E}_1 + \mathcal{R}^{-1}P\mathcal{E}_2 + \cdots + \mathcal{R}^{-1}P\mathcal{E}_m$$

$$= \mathcal{J}_1 + \mathcal{J}_2 + \cdots + \mathcal{J}_m .$$

Incidence Matrices

- Consider a circuit with n-loops. Define

  E_{ab} = the total generators shared by loop a and b

  R_{ab} = the total resistance shared by loop a and b

  I_a = the loop current for loop a

  $$\delta_{ab} = \begin{cases} 1 & , \text{if } a=b \\ 0 & , \text{if } a\neq b. \end{cases}$$

  and note that $E_{ab} = E_{ba}$ and $R_{ab} = R_{ba}$
- Can we write $\mathcal{R}_{ab}$ in terms of R_{ab}?
- Note that the equation for loop a reads:

$$\sum_{b} E_{ab} = \sum_{b\neq a} R_{ab}(I_a - I_b) + R_{aa} I_a$$

The (right-hand-side can be rewritten as:

$$\begin{aligned} RHS &= \sum_{b\neq a} R_{ab}(I_a - I_b) + R_{aa} I_a \\ &= I_a \sum_{b} R_{ab} - \sum_{b\neq a} R_{ab} I_b \\ &= I_a \sum_{\gamma} R_{a\gamma} - \sum_{b} R_{ab}(1-\delta_{ab}) I_b = \\ &= \sum_{b} \left[\delta_{ab} \sum_{\gamma} R_{b\gamma} \right] I_b - \sum_{b} R_{ab}(1-\delta_{ab}) I_b = \end{aligned}$$

$$= \sum_b \left[\delta_{ab} \sum_\gamma R_{b\gamma} - R_{ab}(1-\delta_{ab}) \right] I_b =$$

$$= \sum_b \mathcal{R}_{ab} I_b$$

therefore

$$\boxed{\mathcal{R}_{ab} = \delta_{ab} \sum_\gamma R_{b\gamma} - R_{ab}(1-\delta_{ab})}$$

<u>example</u>

For $n=2$: two loops

$$\mathcal{R}_{11} = \delta_{11} \sum_\gamma R_{1\gamma} - R_{11}(\cancelto{0}{1-\delta_{11}}) = R_{11} + R_{12}$$

$$\mathcal{R}_{12} = \cancelto{0}{\delta_{12}} \sum_\gamma R_{1\gamma} - R_{12}(1-\delta_{12}) = -R_{12}$$

$$\mathcal{R}_{21} = \cancelto{0}{\delta_{21}} \sum_\gamma R_{2\gamma} - R_{21}(1-\delta_{21}) = -R_{21} = -R_{12}$$

$$\mathcal{R}_{22} = \delta_{22} \sum_\gamma R_{2\gamma} - R_{22}(\cancelto{0}{1-\delta_{22}}) = R_{21} + R_{22}$$

$$= R_{12} + R_{22}$$

thus $$\mathcal{R} = \begin{bmatrix} R_{11} + R_{12} & -R_{12} \\ -R_{12} & R_{22} + R_{12} \end{bmatrix}$$

- It is not always true that $\mathcal{R}$ has an inverse.
  e.g. when $R_{11}=0$ and $R_{22}=0$
  then $\det(\mathcal{R})=0$
  This situation indicates a <u>short-circuit.</u>

- The incidence matrix can be of use if you want to write a computer program that can solve arbitrary circuits.

EXERCISE

④ Derive the relation between $\mathcal{R}$ and the incidence matrix R for the general $n=3$ circuit (3 loops). Then calculate and simplify the determinant $\det(\mathcal{R})$. Under what conditions is $\mathcal{R}$ singular?

▼ Least-squares fit

- Consider a set of data points
  $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
  that approximately fall upon a line
  $(\ell) : y = ax + b$.

- Want to find the best possible values for a and b.

▶ Solution

Define

$$\begin{aligned} S_x &= x_1 + x_2 + \cdots + x_n \\ S_y &= y_1 + y_2 + \cdots + y_n \\ S_{xx} &= x_1^2 + x_2^2 + \cdots + y_n^2 \\ S_{yy} &= y_1^2 + y_2^2 + \cdots + y_n^2 \end{aligned}$$

and $S_{xy} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$.

We estimate the error in the line fit by calculating

$$E(a,b) = \sum_{k=1}^{n} (y_k - (ax_k + b))^2$$

To minimize $E(a,b)$ we calculate the partial derivatives with respect to a and b :

$$\frac{\partial E(a,b)}{\partial a} = \frac{\partial}{\partial a} \sum_{k=1}^{n} [y_k - (ax_k + b)]^2 =$$

$$= \sum_{k=1}^{n} \left\{ \frac{\partial}{\partial a} [y_k - (ax_k + b)]^2 \right\} =$$

$$= \sum_{k=1}^{n} \left\{ 2[y_k - (ax_k + b)](-x_k) \right\} =$$

$$= \sum_{k=1}^{n} [-2x_k y_k + 2ax_k^2 + 2bx_k]$$

$$= -2\sum_{k=1}^{n} x_k y_k + 2a\sum_{k=1}^{n} x_k^2 + 2b\sum_{k=1}^{n} x_k$$

$$= -2S_{xy} + 2aS_{xx} + 2bS_x$$

and

$$\frac{\partial E(a,b)}{\partial b} = \frac{\partial}{\partial b} \sum_{k=1}^{n} [y_k - (ax_k + b)]^2 =$$

$$= \sum_{k=1}^{n} \left\{ \frac{\partial}{\partial b} [y_k - (ax_k + b)]^2 \right\} =$$

$$= \sum_{k=1}^{n} \left\{ 2[y_k - (ax_k + b)](-1) \right\} =$$

$$= \sum_{k=1}^{n} [-2y_k + 2ax_k + 2b]$$

$$= -2\sum_{k=1}^{n} y_k + 2a\sum_{k=1}^{n} x_k + 2nb$$

$$= -2S_y + 2aS_x + 2nb$$

At the minimum we have

$$\begin{cases} \dfrac{\partial E(a,b)}{\partial a} = 0 \\ \dfrac{\partial E(a,b)}{\partial b} = 0 \end{cases} \Leftrightarrow \begin{cases} -2 S_{xy} + 2a S_{xx} + 2b S_x = 0 \\ -2 S_y + 2a S_x + 2n b = 0 \end{cases}$$

$$\Leftrightarrow \begin{bmatrix} S_{xx} & S_x \\ S_x & n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} S_{xy} \\ S_y \end{bmatrix}$$

$$D_{a} = \begin{vmatrix} S_{xx} & S_x \\ S_x & n \end{vmatrix} = n S_{xx} - [S_x]^2$$

$$D_a = \begin{vmatrix} S_{xy} & S_x \\ S_y & n \end{vmatrix} = n S_{xy} - S_x S_y$$

$$D_b = \begin{vmatrix} S_{xx} & S_{xy} \\ S_x & S_y \end{vmatrix} = S_{xx} S_y - S_{xy} S_x$$

consequently,

$$a = \frac{n S_{xy} - S_x S_y}{n S_{xx} - [S_x]^2}$$

$$b = \frac{S_{xx} S_y - S_{xy} S_x}{n S_{xx} - [S_x]^2}$$

- Will we always have
$$D = n S_{xx} - [S_x]^2 \neq 0 \ ?$$

<u>Answer:</u> We use the Lagrange identity:

$$\left(\sum_{k=1}^{n} a_k^2\right)\left(\sum_{k=1}^{n} b_k^2\right) - \left(\sum_{k=1}^{n} a_k b_k\right)^2 =$$
$$= \frac{1}{2}\sum_{k=1}^{n}\sum_{\ell=1}^{n} \begin{vmatrix} a_k & b_k \\ a_\ell & b_\ell \end{vmatrix}^2$$

It follows that

$$D = n S_{xx} - [S_x]^2 =$$
$$= (1^2+1^2+\cdots+1^2)(x_1^2+x_2^2+\cdots+x_n^2) -$$
$$(1x_1+1x_2+\cdots+1x_n)^2 =$$
$$= \frac{1}{2}\sum_{k=1}^{n}\sum_{\ell=1}^{n} \begin{vmatrix} 1 & x_k \\ 1 & x_\ell \end{vmatrix}^2 =$$
$$= \frac{1}{2}\sum_{k=1}^{n}\sum_{\ell=1}^{n} (x_k - x_\ell)^2 \geq 0$$

As long as at least two of $x_1, x_2, \ldots, x_n$ are not equal to each other, we see that $D > 0 \Rightarrow D \neq 0$.

- How do we know that the solution we found is really a minimum and not a maximum?

<u>Answer</u>: A sufficient condition for a minimum is that

$$\frac{\partial^2 E(a,b)}{\partial a^2} > 0 \qquad (1) \qquad , \text{and}$$

$$M(a,b) = \frac{\partial^2 E(a,b)}{\partial a^2}\frac{\partial^2 E(a,b)}{\partial b^2} - \left[\frac{\partial^2 E(a,b)}{\partial a \partial b}\right]^2$$

$$> 0 \qquad (2)$$

(1) ⟶ minimum in a direction

(2) ⟶ same currature in all directions.

Recall that

$$\frac{\partial E(a,b)}{\partial a} = -2 S_{xy} + 2a S_{xx} + 2b S_x$$

$$\frac{\partial E(a,b)}{\partial b} = -2 S_y + 2a S_x + 2nb$$

It follows that

$$\frac{\partial^2 E(a,b]}{\partial a^2} = 2 S_{xx} \ , \ \frac{\partial^2 E(a,b)}{\partial b^2} = 2n \ , \text{and}$$

$$\frac{\partial^2 E(a,b)}{\partial a \partial b} = 2 S_x$$

and thus for eq. (1):

$$\frac{\partial^2 E(a,b)}{\partial a^2} = 2S_{xx} = 2\sum_{k=1}^{n} x_k^2 > 0$$

if at least one $x_k \neq 0$.

$$\begin{aligned} M(a,b) &= (2S_{xx})(2n) - (2S_x)^2 = \\ \text{!!!} \quad &= 4(nS_{xx} - (S_x)^2) \\ &= 4D > 0 \leftarrow \text{if at least one} \end{aligned}$$

$x_k - x_l \neq 0$ for $k \neq l$.

- Note the amusing relationship between $M(a,b)$ and the determinant D!!

EXERCISES

⑤ Find the least-square fit line given the data

a) $(-1,-\lambda), (0,0), (1, \lambda+\Delta\lambda)$
b) $(0,0), (1,1), (2, 2+\varepsilon)$
c) $(1,1), (a,a), (2, 2+a)$

⑥ Confirm that given two data points (x_1, y_1) and (x_2, y_2) the least-square fit line corresponds to the line that does in fact go through the two data points (i.e. you have a perfect fit). You will find

$$a = \frac{y_1 - y_2}{x_1 - x_2} \longleftarrow \text{slope}$$

$$b = y_1 - a x_1$$

which corresponds to the line

$$(\ell): \; y - y_1 = a(x - x_1)$$

⑦ Suppose that your data points are not exact and you are given instead

$$(x_k \pm \sigma_{x_k}, y_k \pm \sigma_{y_k}) \;, \; k = 1, 2, 3, \ldots, n$$

You may still calculate a, b as usual but because your data is not exact there will be some error in your predicted a and b. So we want $a \pm \sigma_a$, and $b \pm \sigma_b$.

It can be shown that

$$\boxed{\sigma_a{}^2 = \sum_{k=1}^{n} \left(\frac{\partial a}{\partial x_k}\right)^2 \sigma_{x_k}^2 + \sum_{k=1}^{n} \left(\frac{\partial a}{\partial y_k}\right)^2 \sigma_{y_k}^2}$$

$$\boxed{\sigma_b{}^2 = \sum_{k=1}^{n} \left(\frac{\partial b}{\partial x_k}\right)^2 \sigma_{x_k}^2 + \sum_{k=1}^{n} \left(\frac{\partial b}{\partial y_k}\right)^2 \sigma_{y_k}^2}$$

Calculate the derivatives

$$\frac{\partial a}{\partial x_k}, \quad \frac{\partial a}{\partial y_k}, \quad \frac{\partial b}{\partial x_k}, \quad \frac{\partial b}{\partial y_k}$$

↳ Obviously, given these derivatives we can write a computer program that finds σ_a and σ_b in addition to a and b from the data.

Appendices

Programming with Matlab

Eleftherios Gkioulekas
Mathematical Sciences Computing Center
University of Washington

December, 1996

1 Starting Matlab

Matlab is an interactive tool that includes facilities for dealing with numerical analysis, matrix computation, signal processing and graphics. It is meant to be used to understand and test mathematical concepts interactively before coding in a real programming language.

Throughout this tutorial, we will give you an overview of various things and refer you to Matlab's on-line help for more information. The best way to learn is by experimentation, and the best way this tutorial can help you is by telling you the basics and giving you directions towards which you can take off exploring on your own.

To start Matlab type `matlab` on the shell prompt. You get a greeting on your screen, then a window pops up and goes away [1] and finally you get a `>>` prompt. At this point you can start typing in commands on interactive mode or you can quit by typing `quit` at the prompt. You can also navigate around the filesystem, execute the Matlab programs you have written, and get online help. If you want to log your what you do throughout your matlab session type

```
>> diary filename
```

where in `filename` you put the filename you want to use for your log. To stop logging type

```
>> diary off
```

Moreover you can get on-line help by typing

```
>> help
```

on the prompt. A list of topics appears. You can then type `help` again followed with the name of the topic and get more information about what you want to know. For example suppose you type:

```
>> help
[..lots of stuff..]
Matlab/matfun        -  Matrix functions - numerical linear algebra.
[..lots of other stuff..]
```

[1] The reason this happens is because Matlab tries to see if you have an X display. It can do that without popping up a window, but then you wouldn't be as impressed

There you go! Say you want to learn more about the matrix stuff. You type

```
>> help matfun

 Matrix functions - numerical linear algebra.

 Matrix analysis.
   cond        - Matrix condition number.
   norm        - Matrix or vector norm.
[etc...etc...]
```

and you are given a list of commands that are available to you for working with matrices. Then, to learn about the `cond` command you type

```
>> help cond
```

and you will be told how to use it to compute condition numbers. For more information about getting help try:

```
>> help help
```

Another way of getting help is with the `lookfor` command. Suppose you want to see what Matlab can do with “eigenvalues”. Typing:

```
>> lookfor eigenvalue
```

will return to you a list of commands in which the word “eigenvalue” occurs. Note that this command is not very artificially intelligent. If you are looking for your son, typing

```
>> lookfor son
```

will not give you what you want

If you are on a machine that can run a web browser, try also:

```
>> doc
```

Finally, if Matlab ever does something you don’t understand type:

```
>> why
```

for a succinct explanation.

Like we said, at the prompt you can also execute your programs. So, let’s do that!. Bring up an editor [2] and type in the following Matlab program:

[2] On a Unix system you can use the following editors: `emacs`, `vi` and if available `pico`. On DOS there is `edit`. Other systems have (or are supposed to have) a **`text editor`**. Warning: You can NOT use a word-processor to type in Matlab programs, unless the word processor allows you to save your document as pure text.

```
%
% These are comments. In Matlab you can add comments with a % character
% This is the standard hello world program

disp('Hello world!');
```

Save the program with the filename `hello.m`. You want all your Matlab programs to be saved with the extension `.m` at the end. Then start Matlab *under the same directory where you saved the file* and type `hello` at the Matlab prompt. Then you should see the following:

```
>> hello
Hello world!
>>
```

When you terminate a Matlab command with a semicolon, the command will execute silently. When you don't, the command will print back a response. In a matlab program, the former behaviour is desirable. When using Matlab interactively you may want to see these responses. As a general rule of thumb, when you write Matlab programs terminate every statement with a semi-colon and produce the output of interest by invoking the commands whose job is to print things. Such a command would be `disp` for example, which will print `hello` even though it is being "silenced" with a semicolon. We will learn about another printing command later on.

Notice that within the Matlab environment you don't have to type any funny commands to *load* your program. When you start Matlab from your shell, Matlab will load *all* the `*.m` files that happen to be under the present working directory at startup time. So, all you have to do when the prompt shows up is to type the name of the program (*without the* `*.m` *extension*) and it will get executed. Note that Matlab will only look for files with the `*.m` extension, so you are forced to use it. There are no work-arounds for this. Still, Matlab has a provision for the situation where your files are scattered in more than one directory. You can use the Unix `cd, ls, pwd` commands to navigate in the file system and change your current working directory. Also, if you want to be able to have access at the files on two or more seperate directories *simultaneously* type

```
>> help path
```

for more information.

2 Matlab variables

Matlab has three basic data types: strings, scalars and matrices. Arrays are just matrices that have only one row. Matlab has also lots of built-in functions to work with these things. You have already seen the `disp` function in our hello-world program.

Starting with strings, you can assign a string to a variable like this:

```
name = 'Indiana Jones';
```

Note that it is a syntax error to quote the string with anything other than the forward quote marks. So, the following are **wrong!**

```
name = "Indiana Jones";      wrong!
name = ‘Indiana Jones‘;      wrong!
```

In a Matlab program you can prompt the user and ask him to enter in a string with the `input` command:

```
% This is a rather more social program
%
yourname = input('Hello! Who are you? ','s');
dadname  = input('What's your daddy name? ','s');
fprintf(1,'Hail oh %s son of %s the Great! \n',yourname,dadname);
```

The `input` command takes two arguments. The first argument is the string that you want the user to be prompted with. You could stick in a variable instead of a fixed string if you wanted to. The second argument tells Matlab to expect the user to enter a string. If you omit the second argument, then Matlab will be expecting a number, and upon you entering your name, Matlab will complain. Finally, it *returns* the value that the user enters, and that value is passed on through assignment to the variable `yourname`.

The `fprintf` command gives you more flexibility in producing output than the `disp` command. There is **a lot** to learn about `fprintf` so please type `help fprintf` to learn all you need to know. Briefly, `fprintf` takes two or three or more arguments. The first argument is a *file descriptor*. File descriptors are integers that reference places where you can send output and receive input from. In Matlab, file descriptor 1 is what you use when you want to send things to the screen. The terminology you may hear is that file descriptor 1 sends things to the *standard output*.

The rest of the arguments depend on what you want to print. If all you want to print is a fixed string, then you just put that in as your second argument. For example:

```
fprintf(1,'Hello world!\n');
```

The \n sequence will switch you over to the next line. `disp` will do this automatically, in `fprintf` you must explicitly state that you wish to go to a new line. This is a feature, since there may be situations where you do *not* want to go to a new line.

If you want to print the values of variables interspersed with your string then you need to put appropriate markers like `%s` to indicate where you want your variables to go. Then, in the subsequent arguments you list the variables in the appropriate order, making sure to match the markers. There are many markers and the Matlab online help will refer you to a C manual. The most commonly used markers are the following:

`%s`	Strings
`%d`	Integers (otherwise you get things like 5.0000)
`%g`	Real numbers in scientific notation.

In our example above, we just used `%s`. You will see further examples later on.

Note that if you merely want to print a variable, it is better to use `disp` since it will format it for you. `fprintf` is more useful for those occasions where you want to do the formatting yourself as well as for sending things to a file.

Scalars can be assigned, inputed and printed in a similar fashion. Here is an example:

```
% yet another one of these happy programs
age = input('Pardon for asking but how old are you?');
if (age < 75)
 life_left = 365.25*24*(75 - age);
 fprintf(1,'You have %g hours left of average life expectancy.\n',life_left);
else
 fprintf(1,'Geez! You are that old?!\n');
end
fprintf(1,'Live long and prosper!\n');
```

Note the following:

- String and numeric variables look the same. You don't have to declare the type of the variable anywhere. Matlab will make sure to do *the right thing* (tm).
- When we use `input` to get the value of a numeric variable we omit the second `'s'` argument. This way, Matlab will do error-checking and complain if you entered something that's not a number.
- You can use `fprintf` to print numeric variables in a similar fashion, but you got to use the `%g` marker. If you are printing an integer you must use the `%d` marker, otherwise Matlab will stick in a few zeroes as decimal places to your integer. It is obvious that you can mix strings and numbers in an `fprintf` command, so long as you don't mix up the order of the variables listed afterwards.
- On line

  ```
  life_left = 365.25*24*(75 - age);
  ```

  we see how you can do simple computations in Matlab. It's very similar to C and Fortran and to learn more about the operators you have available type

  ```
  >> help ops
  >> help relops
  ```

- Finally, we have an example of an `if` statement. We will talk of that more later. The meaning should be intuitively obvious.

In addition to ordinary numbers, you may also have complex numbers. The symbols `i` and `j` are reserved for such use. For example you can say:

```
z = 3 + 4*i;
```

or

```
z = 3 + 4*j;
```

where i and j represent $\sqrt{-1}$. If you are already using the symbols i and j as variables, then you can get a new complex unit and use it in the usual way by saying:

```
ii = sqrt(-1);
z = 3 + 4*ii;
```

3 Arrays in Matlab

Next we talk about arrays. In Matlab arrays are dynamic and they are indexed from 1. You can assign them element by element with commands like:

```
a(1) = 23;
a(2) = input('Enter a(2)');
a(3) = a(1)+a(2);
```

It is a syntax error to assign or refer to a(0). This is unfortunate since in some cases the 0-indexing *is* more convenient. Note that you don't have to initialize the array or state it's size at any point. The array will make sure to grow itself as you index higher and higher indices.

Suppose that you do this:

```
a(1) = 10;
a(3) = 20;
```

At this point, a has grown to size 3. But a(2) hasn't been assigned a value yet. In such situations, during growth any unset elements are set to zero. It is good programming practice however not to depend on this and always initialize all the elements to their proper values.

Notice that for the sake of efficiency you might not like the idea of growing arrays. This is because every time the array is grown, a new chunk of memory must be allocated for it, and contents have to be copied. In that case, you can set the size of the array by initializing it with the zeros command:

```
a = zeros(100);
```

This will set a(1),a(2),...,a(100) all equal to zero. Then, so long as you respect these boundaries, the array will not have to be grown at any point.

Here are some other ways to make assignments to arrays:

```
x = [3 4 5 6];
```

will set x equal to an array of 4 values. You can recursively add elements to your array x in various ways if you include x on the right hand side. For example, you can make assignments like

```
x = [x 1 2]     % append two elements at end of the array
x = [1 2 x 3 ] % append two elements at front, one at back
```

How about making deletions? Well, first of all notice that we can access parts of the array with the following indexing scheme:

```
y = x(2:4);
```

will return the an array of `x(2), x(3), x(4)`. So, if you want to delete the last element of the array, you just have to find the size of the array, which you can do with the `size` command.

Yet another way to setup arrays is like this:

```
x = 3 : 1 : 6;
```

This will set x equal to an array of equidistant values that begin at 3, end at 6 and are separated from each other by steps of 1. You can even make backwards steps if you provide a negative stepsize, like this:

```
x = 6 : -1 : 3;
```

It is common to set up arrays like these when you want to plot a function whose values are known at equidistant points.

Finally, to conclude, you may want to know how to load arrays from files. Suppose you have a file that contains a list of numbers separated with carriage returns. These numbers could be the values of a function you want to plot on known values of x (presumably equidistant). You want to load all of these numbers on a vector so you can do things to them. Here is a demo program for doing this:

```
filename = input('Please enter filename:','s');
fd = fopen(filename);
vector = fscanf(fd,'%g',inf);
fclose(fd);
disp(vector);
```

Here is how this works:

- The first line, prompts the user for a filename.

- The `fopen` command will open the file for reading and return a file descriptor which we store at variable `fd`.

- The `fscanf` command will read in the data. You really need to read the help page for `fscanf` as it is a very useful command. In principle it is a little similar to `fprintf`. The first argument is the file descriptor from which data is being read. The second argument tells Matlab, what kind of data is being read. The `%g` marker stands for real numbers in scientific notation. Finally the third argument tells Matlab to read in the entire file in one scoop. Alternatively you can stick in an integer there and tell Matlab to load only so many numbers from the file.

- The `fclose` command will close the file descriptor that was opened.

- Finally the `disp` command will show you what has been loaded. At this point you could substitute with somewhat more interesting code if you will.

Another common situation is data files that contain pairs of numbers separated by carriage returns. Suppose you want to load the first numbers onto one array, and the second numbers to another array. Here is how that is done:

```
filename = input('Please enter filename: ','s');
fd = fopen(filename);
A  = fscanf(fd,'%g %g\n',[2,inf]);
x  = A(1,:);
y  = A(2,:);
fclose(fd);
disp('Here comes x:'); disp(x);
disp('Here comes y:'); disp(y);
```

Again, you need to read the help page for `fscanf` to understand this example better. You can use it in your programs as a canned box until then. What we do in this code snippet essentially is to load the file into a two-column matrix, and then extract the columns into vectors. Of course, this example now leads us to the next item on the agenda: matrices.

4 Matrices in Matlab

In Matlab, arrays are matrices that have only one row. Like arrays, matrices can be defined element by element like this:

```
a(1,1) = 1; a(1,2) = 0;
a(2,1) = 0; a(2,2) = 1;
```

Like arrays, matrices grow themselves dynamically as needed when you add elements in this fashion. Upon growth, any unset elements default to zero just like they do in arrays. If you don't want that, you can use the `zeros` command to initialize the matrix to a specific size and set it equal to zero, and then take it from there. For instance, the following example will create a zero matrix with 4 rows and 5 columns:

```
A = zeros(4,5);
```

To get the size of a matrix, we use the `size` command like this:

```
[rows,columns] = size(A);
```

When this command executes, the variable `rows` is set equal to the number of rows and `columns` is set equal to the number of columns. If you are only interested in the number of rows, or the number of columns then you can use the following variants of `size` to obtain them:

```
rows = size(A,1);
columns = size(A,2);
```

Since arrays are just matrices with one row, you can use the `size(array,2)` construct to get hold of the size of the array. Unfortunately, if you were to say:

```
s = size(array);   % wrong!
```

it would be wrong, because this returns both the number of rows and columns and since you only care to pick up one of the two numbers, you pick up the number of rows, which for arrays is always equal to 1. Not what you want!

Naturally, there are a few other ways to assign values to a matrix. One way is like this:

```
A = [ 1 0 0 ; 0 1 0 ; 0 0 1]
```

This will set `A` equal to the 3 by 3 identity matrix. In this notation you list the rows and separate them with semicolons.

In addition to that you can extract pieces of the matrix, just like earlier we showed you how to extract pieces of the arrays. Here are some examples of what you can do:

```
a = A(:,2);         % this is the 2nd column of A
b = A(3,:);         % this is the 3rd row of A
c = A(1:4,3);       % this is a 4 by 1 submatrix of A
d = A(:,[2 4 10]); % this is the 2nd, 4th and 10th columns of A stacked
```

In general, if `v` and `w` are arrays with integer components, then `A(v,w)` is the matrix obtained by taking the elements of A with row subscripts from `v` and column subscripts from `w`. So:

```
n = size(A,2);
A = A(:,n:-1:1);
```

will reverse the columns of A. Moreover, you can have **all** of these constructs appear on the left hand side of an assignment and Matlab will do the right thing. For instance

```
A(:,[3 5 10]) = B(:,1:3)
```

replaces the third, fifth and tenth columns of A with the first three columns of B.

In addition to getting submatrices of matrices, Matlab will allow you to put together block matrices from smaller matrices. For example if `A,B,C,D` are a square matrices of the same size, then you can put together a block matrix like this:

```
M = [ A B ; C D ]
```

Finally, you can get the transpose of matrix by putting a `'` mark next to it. For example:

```
A = [ 2 4 1 ; 2 1 5 ; 4 2 6];
Atrans = A';
```

Matlab provides functions that return many special matrices. These functions are listed in Figure 1 and we urge you to look these up with the `help` command and experiment.

To display the contents of matrices you can simply use the `disp` command. For example, to display the 5 by 5 Hilbert matrix you want to say:

`zeros`	Returns the zero matrix
`ones`	Returns a matrix in which all entries are set equal to one
`rand`	A matrix with uniformly distributed random elements
`randn`	A matrix with normally distributed random elements
`eye`	The identity matrix
`compan`	Computes the companion matrix
`diag`	Extract one of the diagonals of a matrix
`gallery`	Returns a couple of small test matrices. See help page.
`hadamard`	Returns a Hadamard matrix of any order where N, N/12 or N/20 is a power of 2
`hilb`	Returns the Hilbert matrices
`invhilb`	Returns the inverse of Hilbert matrices
`pascal`	Returns Pascal's triangle
`toeplitz`	Returns Toeplitz matrices
`vander`	Returns Vandermonde matrices.

Figure 1: Matrix commands

```
>> disp(hilb(5))
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

The Hilbert matrix is a famous example of a badly conditioned matrix. It is also famous because the exact inverse of it is known analytically and can be computed with the `invhilb` command:

```
>> disp(invhilb(5))
          25        -300        1050       -1400         630
        -300        4800      -18900       26880      -12600
        1050      -18900       79380     -117600       56700
       -1400       26880     -117600      179200      -88200
         630      -12600       56700      -88200       44100
```

This way you can interactively use these famous matrices to study concepts such as ill-conditioning.

5 Matrix/Array Operations

- **Matrix addition/subtraction**: Matrices can be added and subtracted like this:

  ```
  A = B + C;
  A = B - C;
  ```

  It is necessary for both matrices `B` and `C` to have the same size. The exception to this rule is when adding with a scalar:

```
A = B + 4;
```

In this example, all the elements of `B` are increased by 4 and the resulting matrix is stored in `A`.

- **Matrix multiplication**: Matrices `B` and `C` can be multiplied if they have sizes $n \times p$ and $p \times m$ correspondingly. The product then is evaluated by the well-known formula

  $$A_{ij} = \sum_{k=1}^{p} B_{ik} C_{kj},\ \forall i \in \{1, \ldots, n\},\ \forall j \in \{1, \ldots, m\}$$

  In Matlab, to do this you say:

  ```
  A = B*C;
  ```

  You can also multiply all elements of a matrix by a scalar like this:

  ```
  A = 4*B;
  A = B*4;  % both are equivalent
  ```

  Since vectors are matrices that are $1 \times n$, you can use this mechanism to take the dot product of two vectors by transposing one of the vectors. For example:

  ```
  x = [2 4 1 5 3];
  y = [5 3 5 2 9];
  p = x'*y;
  ```

  This way, `x'` is now a $n \times 1$ "matrix" and `y` is $1 \times n$ and the two can be multiplied, which is the same as taking their dot product.

  Another common application of this is to apply $n \times n$ matrices to vectors. There is a catch though: In Matlab, vectors are defined as matrices with one row, i.e. as $1 \times n$. If you are used to writing the matrix product as Ax, then you have to transpose the vector. For example:

  ```
  A = [1 3 2; 3 2 5; 4 6 2];  % define a matrix
  x = [2 5 1];                % define a vector
  y = A*x;                    % This is WRONG!
  y = A*x';                   % This is correct :)
  ```

- **Array multiplication:** This is an alternative way to multiply *arrays*:

  $$C_{ij} = A_{ij} B_{ij},\ \forall i \in \{1, \ldots, n\},\ \forall j \in \{1, \ldots, m\}$$

  This is not the traditional matrix multiplication but it's something that shows up in many applications. You can do it in Matlab like this:

```
C = A.*B;
```

- **Array division:** Likewise you can divide arrays in matlab according to the formula:

  $$C_{ij} = \frac{A_{ij}}{B_{ij}},\ \forall i \in \{1,\dots,n\},\ \forall j \in \{1,\dots,m\}$$

  by using the `./` operator like this:

  ```
  C = A./B;
  ```

- **Matrix division:** There are two *matrix division* operators in Matlab: `/` and \\. In general

  `X = A\B` is a solution to `A*X = B`
  `X = A/B` is a solution to `X*A = B`

  This means that `A\B` is defined whenever `B` has as many rows as `A`. Likewise `A/B` is defined whenever `B` has as many columns as `A`. If `A` is a square matrix then it is factored using Gaussian elimination. Then the equations `A*X(:,j) = B(:,j)` are being solved for every column of `B`. The result is a matrix with the same dimensions as `B`. If `A` is not square, it is factored using the Householder orthogonalization with column pivoting. Then the corresponding equations will be solved in the least squares fit sense. Right division `A/B` in Matlab is computed in terms of left division by `A/B = (A’\B’)’`. For more information type

  ```
  >> help slash
  ```

- **Matrix inverse:** Usually we are not interested in matrix inverses as much as applying them directly on vectors. In these cases, it’s best to use the matrix division operators. Nevertheless, you can obtain the inverse of a matrix if you need it by using the `inv` function. If `A` is a matrix then `inv(A)` will return it’s inverse. If the matrix is singular or close to singular a warning will be printed.

- **Matrix determinants:** Matrix determinants are defined by

  $$\det(A) = \sum_{\sigma \in S_n} \left\{ \operatorname{sign}(\sigma) \prod_{i=1}^{n} A_{i\sigma(i)} \right\}$$

  where S_n is the set of permutations of the ordered set $(1, 2, \dots, n)$, and $\operatorname{sign}(\sigma)$ is equal to $+1$ if the permutation is even and -1 if the permutation is odd. Determinants make sense only for square matrices and can be computed with the `det` function:

  ```
  a = det(A);
  ```

- **Matrix exponential function:** These are some very fascinating functions. The *matrix exponential* function is defined by

  $$\exp(A) = \sum_{k=0}^{+\infty} \frac{A^k}{k!}$$

  where the power A^k is to be evaluated in the *matrix product* sense. Recall that your ordinary exponential function is defined by

  $$e^x = \sum_{k=0}^{+\infty} \frac{x^k}{k!}$$

  which converges for all x (even when they are complex). It is **not** obvious from this that the corresponding matrix expression also converges. But it does, and the result is the *matrix exponential.* The matrix exponential can be computed in Matlab with the `expm` function. It's usage is as simple as:

  ```
  Y = expm(X);
  ```

  Matrix exponentials show up in the solution of systems of differential equations.

Matlab has a *plethora* of commands that do almost anything that you would ever want to do to a matrix. And we have only discussed a subset of the operations that are permitted with matrices. The following `help` calls should be helpful in exploring what Matlab offers:

```
>> help elmat
>> help matfun
>> help sparfun
```

We will go into a detailed discussion about matrices and linear algebra in Matlab, on a separate tutorial.

6 Flow control in Matlab

So far, we have spent most of our time discussing the data structures available in Matlab, and how they can be manipulated, as well as inputted and outputed. Now we continue this discussion by discussing how Matlab deals with *flow control.*

- **For loops:** In Matlab, a for-loop has the following syntax:

  ```
  for v = matrix
   statement1;
   statement2;
   ....
  end
  ```

The columns of the matrix are stored one at a time in the variable, and then the statements up to the `end` statement are executed. If you wish to loop over the rows of the matrix then you simply transpose it and plug it in. It is recommended that the commands between the `for` statement and the `end` statement are indented by one space so that the reader of your code can visually see that these statements are enclosed in a loop.

In many cases, we use arrays for matrices, and then the for-loop reduces to the usual for-loop we know in languages like Fortran and C. In particular using expressions of the form `X:Y` will effectively make the loop variable be a scalar that goes from `X` to `Y`. Using an expression of the form `start:step:stop` will allow you to loop a scalar from value `start` all the way to value `stop` with stepsize `step`. For instance the Hilbert matrix is defined by the equation:

$$A_{ij} = \frac{1}{i+j+1}$$

If we didn't have the `hilb` command, then we would use *for-loops* to initialize it, like this:

```
N = 10;
A = zeros(N,N);
for i = 1:N
 for j = 1:N
  A(i,j) = 1/(i+j-1);
 end
end
```

The same code can be rewritten more consisely like this:

```
N = 10; A = zeros(N,N);
for i = 1:N , for j = 1:N , A(i,j) = 1/(i+j-1); , end, end
```

Note that the `for` and `end` statements in the long-winded version of the example are being separated by *newlines* and not semicolons. To obtain the consise version, we merely substitute those newlines with *commas*.

- **While loops**: In Matlab while loops follow the following format:

```
while variable
 statement1;
 statement2;
 ....
 statementn;
end
```

where `variable` is almost always a *boolean expression* of some sort. In Matlab you can compose boolean expressions as shown in Figure 2.

Here is an example of a Matlab program that uses the while loop:

a == b	True when a equals b
a > b	True when a is greater than b
a < b	True when a is smaller than b
a <= b	True when a is smaller or equal to b
a >= b	True when a is greater or equal to b
a = b	True when a is not equal to b
a & b	True when both boolean expressions a and b are true
a | b	True when at least one of a or b is true.
a xor b	True only when only one of a or b is true.
a	True when a is false.

Figure 2: Some boolean expressions in Matlab

```
n = 1;
while prod(1:n) < 1.e100
 n = n + 1;
end
disp(n);
```

This program will display the first integer for which $n!$ is a 100-digit number. The `prod` function takes an array (or matrix) as argument and returns the product of it's elements. In this case, `prod(1:n)` is the factorial $n!$.

- **If and Break statements:** The simplest way to set-up an `if` branch is like this:

```
if variable
 statement1;
 ....
 statementn;
end
```

The statements are executed only if the real part of the `variable` has all non-zero elements. [3] Otherwise, the program continues with executing the statements right after the `end` statement. The variable is usually the result of a boolean expression. The most general way to do if-branching is like this:

```
if variable
 statement1;
 ......
```

[3] Note that in the most general case, `variable` could be a complex number. The `if` statement will only look into its real part

```
 statementn;
elseif variable2
 statement1;
 ......
 statementn;
[....as many elseifs as you want...]
else
 statement1;
 ......
 statementn;
end
```

In this case, if `variable` is true, then the statements right after it will execute until the first `else` or `elseif` (whichever comes first), and then control will be passed over to the statements after the `end`. If `variable` is not true, then we check `variable2`. Now, if that one is true, we do the statements following thereafter until the next `else` or `elseif` and when we get there again we jump to `end`. Recursively, upon consecutive failures we check the next `elseif`. If all the `elseif` variables turn out to be false, then we execute the statements after the `else`. Note that the `elseifs` and/or the `else` can be omitted all together.

Here is a general example that illustrates the last two methods of flow control.

```
% Classic 3n+1 problem from number theory
while 1
 n = input('Enter n, negative quits. ');
 if n <= 0, break, end
 while n > 1
  if rem(n,2) == 0 , n = n/2;
  else, n = 3*n+1;
  end
 end
end
```

This example involves a fascinating problem from number theory. Take any positive integer. If it is even, divide it by 2; if it is odd, multiply it by 3 and add 1. Repeat this process until your integer becomes a 1. The fascinating unsolved problem is: Is there any integer for which the process does not terminate? The conjecture is that such integers do not exist. However nobody has managed to prove it yet.

The `rem` command returns the remainder of a Euclidean division of two integers. (in this case `n` and 2)

7 Functions in Matlab

Matlab has been written so that it can be extended by the users. The simplest way to do that is to write *functions* in Matlab code. We will illustrate this with an example: suppose you want to create a function called `stat` which will take an array and return it's mean and standard deviation. To do that you must create a file called `stat.m`. *The file has to have the same name as the function you are defining, and the function definition is the only thing you can put in that file!.* Then in that file, say the following:

```
function [mean, stdev] = stat(x)
% stat -- mean and standard deviation of an array
%   The stat command returns the mean and standard deviation of the
%   elements of an array. Typical syntax is like this:
%   [mean,dev] = stat(x);
%
% See also: foo, gleep, bork
[m,n] = size(x);
if m == 1
 m = n;
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```

The first line of the file should have the keyword `function` and then the syntax of the function that is being implemented. Notice the following things about the function syntax:

- Functions can have an arbitrary number of arguments. In this case there is only one such argument: `x`. The arguments are being passed by *value*: The variable that the calling code passes to the function is *copied* and the copy is being given to the function. This means that if the function internally changes the value of x, the change will **not** reflect on the variable that you use as argument on your main calling code. Only the copy will be changed, and that copy will be discarded as soon as the function completes it's call.

- Of course it is not desirable to only pass things by value. The function has to communicate some information back to the calling code. In Matlab, the variables on the right hand side, listed in brackets, are also being passed to the function, but this is done **by reference**: The function is not given a *copy* of the variables but it is actually given the variables *themselves*. This means that that any changes made to those variables while inside the function will reflect on the corresponding variables on the calling code. So, if one were to call the function with:

  ```
  a = 1:20;
  m = 0;
  s = 0;
  [m,s] = stat(a);
  ```

then the values of the variables m and s would change after the call to stat.

- The lines afterwords are comments. However, these comments are what will be spit out if you type

  ```
  >> help stat
  ```

  on your prompt. You usually want to make the first comment line be a summary of what the function does because in some instances only the first line will get to be displayed, so it should be complete. Then in the lines afterwords, you can explain how the function is meant to be used.

- After you are done with documentation you type in your usual Matlab code that will implement the function.

The main problem with Matlab function definitions is that you are forced to put every function in a separate file, and are even restricted in what you can call that file. Another thing that could cause you problems is *name collision*: What if the name you choose for one of your functions happens to be the name of an obscure built-in Matlab function? Then, your function will be completely ignored and Matlab will call up the built-in version instead. To find out if this is the case use the which command to see what Matlab has to say about your function.

Many of the examples we saw earlier would be very useful if they were to implemented as functions. For instance, if you commonly use Matlab to manipulate data that come out in (x, y) pairs you can make our earlier example into a function like this:

```
function [x,y] = load_xy(filename)
% load_xy  -- Will allow you to load data stored in (x,y) format
% Usage:
% Load your data by saying
%   [x,y] = load_xy(filename)
% where 'filename' is the name of the file where the data is stored
% and 'x' and 'y' are the vectors where you want the data loaded into
fd = fopen(filename);
A  = fscanf(fd,'%g %g\n',[2,inf]);
x  = A(1,:);
y  = A(2,:);
fclose(fd);
```

You would have to put this in a file called load_xy.m of course. Suppose that after making some manipulations you want Matlab to save your data on file again. One way to do it is like this:

```
function save_xy(filename,x,y)
% save_xy  -- Will let your save data in (x,y) format.
% Usage:
% If x and y are vectors of equal length, then save them in (x,y)
```

```
% format in a file called 'filename' by saying
% save_xy(filename,x,y)
fd = fopen(filename,'w');
A(1,:) = x;
A(2,:) = y;
fprintf(fd,'%g %g\n',A);
fclose(fd);
```

Notice that it is not necessary to use a `for` loop to `fprintf` or `fscanf` the data one by one. This is explained in detail in the on-line help pages for these two commands. In many other cases Matlab provides ways to eliminate the use of `for`-loops and when you make use of them, your programs will generally run faster. A typical example is *array promotion.* Take a very simple function

```
function y = f(x)
```

which takes a number `x` and returns another number `y`. Typical such functions are `sin, cos, tan` and you can always write your own. Now, if instead of a number you plug in an array or a matrix, then the function will be applied on every element of your array or matrix and an array of the same size will be returned. This is the reason why you don't have to specify in the function definition whether `x` and `y` are simple numbers, or arrays in the first place! To Matlab everything is a matrix as far as functions are concerned. Ordinary numbers are seen as `1x1` matrices rather than numbers. You should keep that in mind when writing functions: sometimes you may want to multiply your `x` and `y` with the `.*` operator instead of the `*` to handle array promotion properly. Likewise with division. Expect to be surprised and be careful with array promotion.

Let's look at an example more closely. Suppose you write a function like this:

```
function x = foo(y,z)
x = y+z;
```

Then, you can do the following on the Matlab prompt:

```
>> disp(foo(2,3))
     5
>> a = 1:1:10;
>> b = 1:2:20;
>> disp(foo(a,b))
     2     5     8    11    14    17    20    23    26    29
```

What you will *not* be allowed to do is this:

```
>> a = 1:1:10
>> b = 1:1:20
>> disp(foo(a,b))
??? Error using ==> +
Matrix dimensions must agree.
```

```
Error in ==> /home/lf/mscc/matlab/notes/foo.m
On line 2  ==> x = y+z;
```

The arguments a and b can not be added because they don't have the same size. Notice by the way that we used addition as our example on purpose. We challenge you to try * versus .* and see the effects!

One use of functions is to build complex algorithms progressively from simpler ones. Another use is to automate certain commonly-used tasks as we did in the example of loading and saving (x, y) pairs. Functions do not solve all of the worlds problems, but they can help you a lot and you should use them when you have the feeling that your Matlab program is getting too long and complicated and needs to be broken down to simpler components.

Algorithms with Matlab

Eleftherios Gkioulekas
Mathematical Sciences Computing Center
University of Washington

December, 1996

1 Introduction

Numerical analysis is the branch of mathematics whose goal is to figure out how computers can solve problems in a way that's fast, efficient and accurate. *Linear algebra* is a large part of numerical analysis, because many problems eventually reduce to one of the following linear algebra problems:

- We want to solve a system of linear equations.
- We want to solve an eigenvalue problem.

For this reason, researchers think of these problems as "elementary" the way you would think of addition and multiplication elementary enough to use a calculator. Likewise, researchers will use a "calculator" in some sense of the word. Matlab can serve as such a calculator. Beyond Matlab you have the option of using software libraries. Such libraries exist for most commonly used languages like `C++` (e.g. `LAPACK++`) and `FORTRAN` (e.g. LINPACK, EISPACK). For more information about the latter visit the Netlib repository at `http://www.netlib.org/` or send email to `netlib@ornl.gov` with one line that says `send index`.

In this tutorial we are assuming that you have read and understood the *"Programming with Matlab"* tutorial. The purpose of this tutorial is to review the mathematical concepts of linear algebra to give you a feel for the "big picture" and at the same time show you how you can experiment with these concepts using Matlab. As you will see, these problems are not quite "elementary"

2 Basic concepts

A *complex matrix* $\mathbf{A}$ of size $n \times m$ is a mapping

$$\mathbf{A} : \{1, \ldots, n\} \times \{1, \ldots, m\} \mapsto \mathcal{C}$$

where $\mathcal{C}$ is the set of complex numbers. In other words, a matrix will take two integers i and j such that $1 \leq i \leq n$ and $1 \leq j \leq m$ and return back a complex number. That number we denote with a_{ij} (we use the lowercase letter). The set of all $n \times m$ complex matrices is denoted as $\mathcal{C}^{n\times m}$. If we restrict ourselves to real values only, then we are dealing with *real matrices* and the set of these is denoted as $\mathcal{R}^{n\times m}$. Finally, for the sake of notation, we will denote the set of all integers from 1 to n with the symbol:

$$[n] = \{1, \dots, n\}$$

In Matlab all data structures are matrices. In particular ordinary numbers are 1×1 matrices, vectors (or "arrays" in general) are $1 \times n$ matrices. The *Programming with Matlab* tutorial has covered the basics of how all these data structures are setup in Matlab.

There exist various operations defined for matrices:

Definition 1 *(a) Let* $\mathbf{A}, \mathbf{B} \in \mathcal{C}^{n\times m}$ *be two matrices. The sum and difference of these matrices is:*

$$\begin{aligned} \mathbf{C} = \mathbf{A} + \mathbf{B} &\iff c_{ij} = a_{ij} + b_{ij},\ \forall(i,j) \in [n] \times [m] \\ \mathbf{C} = \mathbf{A} - \mathbf{B} &\iff c_{ij} = a_{ij} - b_{ij},\ \forall(i,j) \in [n] \times [m] \end{aligned}$$

(b) Let $\mathbf{A} \in \mathcal{C}^{n\times m}$ *and* $\mathbf{B} \in \mathcal{C}^{m\times p}$*. The product of these matrices is*

$$\mathbf{C} = \mathbf{AB} \iff c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj},\ \forall(i,j) \in [n] \times [p]$$

(c) The transpose of a matrix $\mathbf{A} \in \mathcal{C}^{n\times m}$ *is:*

$$\mathbf{B} = \mathbf{A}^T \iff b_{ij} = a_{ji},\ \forall(i,j) \in [n] \times [m]$$

(d) The hermitian or complex conjugate transpose is:

$$\mathbf{B} = \mathbf{A}^H \iff b_{ij} = a^*_{ji},\ \forall(i,j) \in [n] \times [m]$$

Moreover, there are the following important classes of matrices that we will see mentioned later on:

Definition 2 *A matrix* $\mathbf{A} \in \mathcal{C}^{n\times n}$ *is:*
(a) symmetric $\iff \mathbf{A} = \mathbf{A}^T$
(b) hermitian $\iff \mathbf{A} = \mathbf{A}^H$
(c) orthogonal $\iff \mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A} = \mathbf{I}$
(d) unitary $\iff \mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A} = \mathbf{I}$
(e) normal $\iff \mathbf{A}\mathbf{A}^H = \mathbf{A}\mathbf{A}^H$

For real matrices, hermitian means the same as symmetric, unitary means the same as orthogonal, and *both* of these distinct classes are normal.

In Matlab matrices that are stored in `A` and `B` can be added or multiplied quite simply by saying:

```
C = A + B;
C = A - B;
C = A * B;
```

The *transpose* and the *hermitian* of a matrix can be obtained by

```
B = A.';      % transpose
B = A';       % hermitian
```

The usual rules apply to these operations except for the following:

- Multiplication is not commutative for all matrices. That is there exist matrices such that

$$\mathbf{AB} \neq \mathbf{BA}$$

- Square matrices don't always have a *multiplicative inverse.* A multiplicative inverse is defined by:

$$\mathbf{B} = \mathbf{A}^{-1} \Longleftrightarrow \mathbf{AB} = \mathbf{BA} = \mathbf{I}$$

The first exception is not very interesting, but the second one is related to one of the two "Big Questions" in linear algebra: Solving linear systems of equations.

3 Matrix inversion is not easy

Suppose that $\mathbf{A} \in \mathcal{C}^{n \times n}$ is a square matrix and $\mathbf{x}, \mathbf{b} \in \mathcal{C}^{n \times 1}$ are vectors (actually, column matrices). If $\mathbf{A}$ and $\mathbf{b}$ are known, then we want to find an $\mathbf{x}$ such that

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

If $\mathbf{A}$ has an inverse then $\mathbf{x}$ is unique and given by:

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

otherwise $\mathbf{x}$ will either not exist, or will not be unique.

In order for a matrix to have an inverse, the matrix has to be square. Given that, the existence of matrix inverses is determined by a quantity called *determinant.* The determinant is defined in terms of *permutations* so we must explain what these are first. A permutation $\sigma \in S_n$ of order n is a bijection that maps:

$$\sigma : [n] \mapsto [n]$$

The set of permutations of order n is written as S_n and it has $n!$ elements. In simple terms, take an ordered set of integers say $(1, 2, 3, 4, 5)$ and reorder it to $(4, 1, 3, 2, 5)$. Our concept of how we reordered an ordered set of things is what is being modeled by permutations. There are as many permutations as there are ways in which we can reorder things. One way to represent permutations is by showing reorderings of $(1, 2, \ldots, n)$. For example the permutations of order 3 are:

$$S_3 = \{(1, 2, 3), (2, 3, 1), (3, 1, 2), (1, 3, 2), (3, 2, 1), (2, 1, 3)\}$$

Permutations have an important property called *parity.* If $\sigma \in S_n$ is a permutation then we define the parity to be equal to:

$$s(\sigma) = \text{sign} \prod_{j=1}^{n-1} \prod_{i=j+1}^{n} (\sigma(i) - \sigma(j))$$

When the parity is $+1$ we talk of *even* permutations. When it is -1 we talk of *odd* permutations. You can verify that the even permutations in S_3 are

$$\{(1,2,3),(2,3,1),(3,1,2)\}$$

and the odd ones are

$$\{(1,3,2),(3,2,1),(2,1,3)\}.$$

Given these definitions, the determinant of a matrix $\mathbf{A} \in \mathcal{C}^{n\times n}$ is defined by:

$$\det(\mathbf{A}) = \sum_{\sigma \in S_n} s(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

Unfortunately, this is not the most useful definition in terms of numerics. There are $n!$ terms to add, and every one of these terms involves n multiplications. This will overwhelm any computer soon enough, not to mention round-off errors. You have probably seen many recursive definitions of the determinant in other courses. Those are not helpful either because they also involve work that increases by $n!$.

Most computations of determinants are based on the following theorem:

Theorem 1 *Let* $\mathbf{A}, \mathbf{B} \in \mathcal{C}^{n\times n}$ *be two square matrices. Then,*

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

We write our matrix as a product of matrices whose determinants are easy to compute and apply the theorem. More on this later.

Now the reason why determinants are important is because of this theorem:

Theorem 2 *Let* $\mathbf{A} \in \mathcal{C}^{n\times n}$ *be a square matrix.*

$$\mathbf{A}^{-1}\text{exists} \Longleftrightarrow \det(\mathbf{A}) \neq 0$$

Notice by the definition that the determinant varies continuously as you vary one of the elements of the matrix. This means that determinants are quite reasonable functions of n^2 variables which has the following implication: Most of the time, the determinant will be quite nonzero and an inverse will exist. Sometimes, the determinant will be very close to zero and an inverse will exist but will be so sensitive to the original matrix that it will be hard to compute. In this case we say we have an *ill-conditioned* matrix. Finally, on very rare matrices the determinant will be *exactly* zero and then the inverse just doesn't exist. In this case we have a *singular* matrix. These last two cases cause trouble from a numerical point of view. To make matters worse, deciding numerically which case is which is not trivial either.

In Matlab computing determinants and inverses may appear innocuously simple. If `A` is your matrix then

```
x = det(A);
```

will compute the determinant and

```
B = inv(A);
```

will return the inverse. However, for ill-conditioned matrices, `inv` will not give you the correct inverse. *Sometimes* Matlab will detect this, but not always.

There exists a class of square matrices $\mathbf{H}^{(n)}$ called the *Hilbert matrices* that are defined by:

$$h_{ij} = \frac{1}{i+j-1}$$

The matlab command `hilb` will return a Hilbert matrix of any size. That is:

```
A = hilb(10);
```

will return a Hilbert matrix of size 10. These matrices have two properties: Inverting them is very sensitive to floating point error but there is an analytic equation that gives their exact inverse! The inverse is given by

$$\tilde{h}_{ij} = \tilde{h}_{ji} = \frac{r_{ij}}{i+j-1}\,, \forall\, 1 \le i \le j \le n$$

where r_{ij} is defined for $j > i$ by the following recurrences:

$$\begin{cases} r_{i,i} = p_i^2 \\ r_{ij} = -\dfrac{(n-j+1)(n+j-1)}{(j-1)^2} r_{i,j-1},\ j > i \end{cases} \qquad \begin{cases} p_0 = n \\ p_i = \dfrac{(n-i+1)(n-i-1)}{(i-1)^2} p_{i-1},\ 1 \le i \le n \end{cases}$$

This computation is being done by the `invhilb` command. For example, for $n = 10$ you would say:

```
A = invhilb(10);
```

We can use these matrices to show you the limitations of the `inv` command. Try the following program:

```
for i = 1:20
 x = max(max(abs(invhilb(i)-inv(hilb(i)))));
 fprintf(1,'For size i = %d deviation is x = %g \n',i,x);
end
```

This program will progressively increase the matrix size and attempt to invert the matrix. Then it will compare it with the exact known inverse and report the worst value of the element by element difference. The output of the program is:

```
For size i = 1 deviation is x = 0
For size i = 2 deviation is x = 3.55271e-15
For size i = 3 deviation is x = 6.82121e-13
For size i = 4 deviation is x = 5.96629e-10
For size i = 5 deviation is x = 1.43918e-08
For size i = 6 deviation is x = 0.000447804
For size i = 7 deviation is x = 0.436005
For size i = 8 deviation is x = 35.4408
For size i = 9 deviation is x = 395134
For size i = 10 deviation is x = 3.657e+08
For size i = 11 deviation is x = 3.17671e+11

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 3.659249e-17

For size i = 12 deviation is x = 2.81767e+14
[....etc....]
```

Notice that the deviation managed to increase all the way to `3.657e+08` before Matlab started giving out warnings. Not good! There are two or three sets of errors involved here:

- The errors caused by representing `hilb(n)`
- The errors caused in the matrix inversion process
- The errors, if any, in representing `invhilb(n)`

It turns out that the first of these, which involves representing fractions like 1/3 and 1/5 in floating-point, is the most significant. This error is propagated throughout the matrix inversion and is significantly amplified.

The moral of the story: Sometimes computers don't answer the question you think you are asking them. You think you are asking the computer to find the inverse. When you use the `inv` function, you are asking it to apply an algorithm which *you* believe will give you the inverse.

Exercise 1 *Write a Matlab function that implements the recurrence formula for the inverse Hilbert matrix. Compare your results with the* `invhilb` *command.*

Exercise 2 *Can we trust the* `det` *function in evaluating the determinant for the Hilbert matrix? Look at the errors in computing:* `det(hilb(n))*det(invhilb(n))` *and* `det(hilb(n)*invhilb(n))` *Which is more accurate? Why?*

4 Algorithms related to solving linear systems

After spending all this time talking about inverses, we need to tell you a little secret: we rarely ever compute inverses explicitly, because even if we did, multiplying them with other vectors or with the original matrix itself is susceptible to many floating point errors. In most problems our main goal is to solve the system

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

and we can do this without computing the inverse. The purpose of our earlier discussion was to alert you that there are difficulties we have to overcome: it's hard to compute determinants, so

Cramer rule is out of the question; matrices can be nasty and matrix inversion can be error-prone. Now we come to the techniques that help us deal with these difficulties.

The techniques are quite involved, and Matlab wants to hide the complexity for the user. Therefore, Matlab uses the backslash operator to do the job

```
x = A\b;
```

This will work provided that the matrix $\mathbf{A}$ is well-behaved. However, in order for Matlab to be flexible, they provide you with further access to particular techniques that you may want to use to do matrix inversions by yourself.

4.1 LU decomposition

One technique is with the *LU decomposition.* The idea here is that we write $\mathbf{A}$ as the product of a lower triangular matrix $\mathbf{L}$ with 1 in the diagonal, and an upper triangular matrix $\mathbf{U}$. Then we reduce our problem to the following two systems of equations:

$$\mathbf{Ax} = \mathbf{b} \Longleftrightarrow \mathbf{LUx} = \mathbf{b} \Longleftrightarrow \begin{cases} \mathbf{Ly} = \mathbf{b} \\ \mathbf{Ux} = \mathbf{y} \end{cases}$$

These systems can be solved directly without having to find the inverse. As a matter of fact, given the LU decomposition of a matrix you can find the inverse of $\mathbf{A}$ by rewritting the equation

$$\mathbf{AA}^{-1} = \mathbf{I}$$

as n linear systems involving the unknown columns of $\mathbf{A}^{-1}$. Also, since the diagonal elements of $\mathbf{L}$ are 1 and since $\mathbf{U}$ is upper triangular, the determinant of $\mathbf{A}$ can be determined by:

$$\det(\mathbf{A}) = \det(\mathbf{L})\det(\mathbf{U}) = \det(\mathbf{U}) = \prod_{i=1}^{n} u_{ii}$$

The work involved in computing the LU decomposition increases by $O(n^3)$ which is a vast improvement over $O(n!)$. In Matlab, the LU decomposition can be computed with the command:

```
[L,U] = lu(A);
```

Unfortunately, there is a wrinkle with this method as well as with the matrix $\mathbf{L}$ returned by Matlab that has to do with the LU decomposition existance theorem. That theorem states:

Theorem 3 *Let $\mathbf{A} \in \mathcal{C}^{n\times n}$ be a square matrix and $\mathbf{A}_k \in \mathcal{C}^{k\times k}$ be the $k \times k$ submatrix of $\mathbf{A}$ containing the upper left part. Then,*

$$\mathbf{A} \text{ has an LU decomposition} \Longleftrightarrow \forall k \in [n] : \det(\mathbf{A}_k) \neq 0$$

There are many cases where the condition of this theorem will almost fail: one of the submatrix determinants will be close to zero. The workaround is to permute the rows and columns of A in such a way so that these determinants turn out okey. When you do that you essentially find the LU decomposition of a new matrix **PA** where **P** is a permutation matrix. Permutation matrices have one 1 located in each row and column and when they are applied to another matrix, they permute the other matrix's rows or columns. Since permutations are easy to undo, **P** is yet-another-matrix that's easy to invert: It's inverse is $\mathbf{P}^{-1} = \mathbf{P}^T$. When you run the matlab function `lu` as quoted above, it will not really return an actual lower triangular matrix in `L`. Instead it will do the decomposition $\mathbf{PA} = \mathbf{LU}$ and *return back* $\mathbf{P}^{-1}\mathbf{L}$ *and* $\mathbf{U}$*!*. If you don't like that, you can call `lu` like this:

```
[L,U,P] = lu(A);
```

Now, **L** will be an actual lower triangular matrix and **P** will be a permutation matrix, and the actual decomposition will be

$$\mathbf{A} = \mathbf{P}^T\mathbf{LU}$$

The solution to our system of linear equations then is:

$$\mathbf{x} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{Pb}$$

Now we have a problem: Matlab doesn't have documented facilities for applying $\mathbf{L}^{-1}$ and $\mathbf{U}^{-1}$ with the well-known back/forward-substitution algorithms. Typically, if you want to solve a system of equations with this method , the backslash operator will do it all in one scoop. However, suppose that you want to solve *many* systems of linear equations in which **A** is the same and only **b** vary. Then, you can have a more efficient program going if do the LU decomposition once and use the `L,U,P` arrays on each different vector `b`. You can write your own function to do that. Then again, if efficiency is such a concern, then perhaps it's time to switch to a programming language.

4.2 QR decomposition

Another technique is the *QR decomposition.* This method will decompose **A** to an *orthogonal* matrix **Q** and a right-triangular matrix **R**. That **Q** is orthogonal means that:

$$\mathbf{Q}^{-1} = \mathbf{Q}^T$$

which makes it easy to invert. Also **R** can be easily inverted being a right-triangular matrix, so by rewriting a linear-systems problem as

$$\mathbf{Rx} = \mathbf{Q}^T\mathbf{b}$$

we can solve it, following the footsteps of the LU technique. You can also compute determinants. It turns out that

$$\det(\mathbf{Q}) = 1$$

so

$$\det(\mathbf{A}) = \det(\mathbf{R}) = \prod_{i=1}^{n} r_{ii}$$

The QR method requires twice as many operations as LU, so LU is more commonly used. However, as you will learn in your numerical methods course, there are many other instances where QR decompositions can come in handy. One application of QR decompositions is in obtaining an orthonormal basis to a vector space. If you think of the columns of **A** and the columns of **Q** as vectors, then both sets of vectors span the same space. The difference is that the vectors obtained through matrix **Q** are orthogonal with one another.

To do a QR decomposition in Matlab call:

```
[Q,R] = qr(A);
```

This will produce two matrices **Q** and **R** such that $\mathbf{A} = \mathbf{QR}$. Matlab provides a few variations to this call:

```
[Q,R,P] = qr(A);
```

will produce a permutation matrix **P**, and return in **Q** and **R** the QR decomposition of **AP**. The permutation matrix is chosen so that the absolute values of the diagonal elements $|r_{ii}|$ is decreasing as i increases.

4.3 Cholesky decomposition

When a matrix $\mathbf{A} \in \mathcal{R}^{n \times n}$ is *symmetric* and *positive definite* it has a more efficient triangular decomposition. Symmetric means that

$$a_{ij} = a_{ji}, \forall\, i, j \in [n]$$

and positive definite means that

$$\mathbf{x}^T \mathbf{A} \mathbf{x} > 0 \forall \mathbf{x} \in \mathcal{R}^{n \times 1}$$

If both conditions are true, then **A** can be decomposed to

$$\mathbf{A} = \mathbf{R}^T \mathbf{R}$$

where **R** is an upper triangular matrix and $\mathbf{R}^T$ is the transpose. In complex matrices, you must use the complex conjugate transpose instead. The Cholesky algorithm will attempt to compute this decomposition. The algorithm can self-detect when it fails, and in practice this ability is used to establish whether symmetric matrices *are* positive definite. In Matlab, you can invoke the Cholesky algorithm with the `chol` command:

```
R = chol(A);
```

If the algorithm fails, Matlab will issue a warning. You can suppress the warning by calling `chol` like this:

```
[R,p] = chol(A)
```

If p is non-zero, then this means that the cholesky algorithm failed and the original matrix is not positive definite. Positive definiteness is an important property. Many theorems in linear algebra are known for positive definite matrices, and many times it is useful to be able to decide numerically whether a symmetric matrix is positive definite. Also, when we *know* that the matrix **A** in a system $\mathbf{Ax} = \mathbf{b}$ is positive definite, we can use this method to solve the system; it would be faster than LU.

4.4 SVD decomposition

To conclude, another very useful decomposition is the SVD decomposition. The theorem behind SVD decompositions states:

Theorem 4 *If* $\mathbf{A} \in \mathcal{R}^{m \times n}$ *then there exist two orthogonal matrices* $\mathbf{U} \in \mathcal{R}^{m \times m}$ *and* $\mathbf{V} \in \mathcal{R}^{n \times n}$ *such that*

$$\mathbf{U}^T \mathbf{A} \mathbf{V} = diag(\sigma_1, \sigma_2, \ldots, \sigma_p) \in \mathcal{R}^{m \times n}$$

where $p = \min\{m, n\}$ *and* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$.

To compute the SVD decomposition of a matrix $\mathbf{A}$ in matlab you use the `svd` function as follows:

```
[U,S,V] = svd(A);
```

where `S` will be a diagonal matrix containing the σ_i. If you only want to look at the σ_i, then do

```
s = svd(A);
```

This call will return an array containing just the σ_i. Given the SVD decomposition, the inverse of a square nonsingular matrix $\mathbf{A}$ can be computed with:

$$\mathbf{A}^{-1} = \mathbf{V} \cdot [diag(1/\sigma_j)] \cdot \mathbf{U}^T$$

and the solution to a linear system $\mathbf{Ax} = \mathbf{b}$ becomes:

$$\mathbf{x} = \mathbf{V} \cdot [diag(1/\sigma_j)] \cdot \mathbf{U}^T \mathbf{b}$$

The reason why SVD is important is because it can diagnose two pathologies: *ill-conditioning* and *singularity.* That you can tell from the *condition number* which is defined by:

$$\text{cond}(\mathbf{A}) = \frac{\max\{\sigma_i\}}{\min\{\sigma_i\}}$$

When the condition number is too large, i.e. close to $1/\epsilon$ where ϵ is the floating point accuracy, then we say that the matrix is *ill-conditioned.* When this happens, merely computing the equation above for the inverse is very susceptible to floating point errors. One remedy is to increase the floating point accuracy to that needed. However, there is a theorem that suggests a better approach. That theorem says that if you take those $1/\sigma_i$ that are really bad and replace them with zero and then do the equation above as if nothing is wrong, then you can get a much more accurate answer than *both* the SVD solution *and* a direct method; accurate in the sense that the residual:

$$r = |\mathbf{Ax} - \mathbf{b}|$$

will be much smaller than the numerical result yielded by a direct method or the ordinary SVD method. It takes some discretion to decide which $1/\sigma_i$ to kill, and whether you get the desirable accuracy in the residual. A recommended tolerance is to eliminate those σ_i such that

$$\sigma_i < \epsilon \cdot \max\{m, n\} \cdot \max\{\sigma_i\}$$

where m and n are the size of $\mathbf{A}$, and ϵ the floating point precision. The matrix $\mathbf{A}^{-1}$ obtained by the expression above in which the small σ_i have been eliminated is called the *pseudoinverse* matrix. You can use the `pinv` command to compute the pseudoinverse using the SVD method. One way to invoke this command is by saying:

```
B = pinv(A);
```

in which case the tolerance mentioned above is used. Alternatively, you can specify your own tolerance `tol` by invoking the `pinv` command like this:

```
B = pinv(A,tol);
```

It is important to understand that the pseudoinverse matrix is *not* really the same as the inverse matrix. The point is that in an ill-conditioned problem, we are going to get a more accurate answer if we use the pseudoinverse, rather than if we attempt to compute the inverse and use that instead.If, in the worst case scenario, the residual we get by this method is still not good enough then your other option is to increase your floating point precision. To do this, assign the variable `eps` in matlab to your desired precision. For example, to get quad precision set:

```
eps = 1e-32;
```

When the condition number is infinite, that is when some $\sigma_i = 0$ exactly, the matrix has no inverse. Unfortunately, numerics being what it is, if you have an actual singular matrix, you probably won't see exact zeroes because of round-off error. So, it takes some discretion to decide whether you are dealing with a singular matrix or an ill-conditioned matrix as well.

Matlab has a special command for estimating condition numbers:

```
c = cond(A);
```

The `cond` will compute the condition number. There are two other algorithms for condition estimation that can be invoked by the commands `condest` and `rcond`. See the help pages for more information.

The orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ can also give us their share of information. If the n columns of $\mathbf{A}$ are vectors spanning a vector space of interest, then the columns of $\mathbf{U}$ will contain an *orthogonal* set of vectors spanning the same space. Nothing special so far, since you can do this with the QR decomposition too. The bonus is that with SVD you can detect whether the dimensionality of the vector space is the same as the amount of vectors that you use to span the space. If any zeroes (or nearly zeroes) occur in one of the σ_i then this means that the vector space was not in fact n-dimensional. To deal with that case, you just eliminate the columns in $\mathbf{U}$ that correspond to the ill-behaved σ_i. The vectors in $\mathbf{U}$ that remain *will be* an orthogonal basis for the vector space.

An equivalent way of stating the previous paragraph is to say that SVD can be used to determine the *rank* of the matrix $\mathbf{A}$. Rank is the dimensionality of the vector space spanned by the columns of a matrix. [1] We say that a square matrix $\mathbf{A} \in \mathcal{R}^{n \times n}$ is full rank if it has rank n.

[1] it doesn't actually make a difference if you pick the rows of the matrix. A theorem tells you that you will get the same number, always

Many theorems in linear algebra apply only on matrices that are full-rank, so from the numerical perspective we want to have a way to diagnose matrices that are not full rank. SVD provides us with a method to do that.

In addition to rank, we can also determine the *null space* of a matrix $\mathbf{A} \in \mathcal{C}^{n\times m}$. The null space is defined to be

$$\text{null}(\mathbf{A}) = \{\mathbf{x} \in \mathcal{C}^{m\times 1} : \mathbf{Ax} = \mathbf{0}\}$$

that is, the set of vectors $\mathbf{x}$ for which $\mathbf{Ax}$ is zero. For non-singular matrices, the null-space consists only of $\mathbf{x} = \mathbf{0}$. For matrices of $\text{rank}(\mathbf{A}) = r < m$ the last $m - r$ columns of $\mathbf{V}$ give an orthonormal basis for the nullspace. Finally for matrices of $\text{rank}(\mathbf{A}) = r > m$ the null space is $\text{null}(\mathbf{A}) = \{\mathbf{0}\}$

These concepts become useful when you are trying to solve linear systems

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

in which $\mathbf{A} \in \mathcal{C}^{n\times m}$ is no-longer a square matrix. In such cases, if $\mathbf{A}$ actually has a null space, to find the set of $\mathbf{x}$ that satisfy the linear system, we find one particular solution $\mathbf{x}_0$. Then full solution set becomes:

$$\{\mathbf{x} \in \mathcal{R}^{m\times 1} : \mathbf{Ax} = b\} = \{\mathbf{x} + \mathbf{x}_0 : \mathbf{x} \in \text{null}(\mathbf{A})\}$$

In practice, we don't care to know the full solution space. Instead we want to find the unique $\mathbf{x}$ that minimizes the residual

$$r(\mathbf{x}) = |\mathbf{Ax} - \mathbf{b}|$$

The backslash operator will solve this problem. To find the $\mathbf{x}$ that minimizes the residual do:

```
x = A\b;
```

Exercise 3 *Write a matlab function which will accept a set of vectors spanning a linear space in the form of a matrix, and optionally a threshold for the σ_i. Your function should return an orthogonal set of vectors that span the same space, but take care to remove the vectors whose σ_i falls bellow the threshold.*

5 Eigenvalue Problems

A completely different type of problem that shows up in many forms is the *eigenvalue problem.* Let $\mathbf{A} \in \mathcal{C}^{n\times n}$ be a square matrix, $\mathbf{x} \in \mathcal{C}^{n\times 1}$ be a vector and $\lambda \in \mathcal{C}$ be a complex number. We want to find the pairs $(\lambda, \mathbf{x})$ that satisfy the linear equation:

$$\mathbf{Ax} = \lambda\mathbf{x}$$

The set of λ for which corresponding $\mathbf{x}$ exist are the *eigenvalues* of the matrix. The $\mathbf{x}$ that correspond to each eigenvalue form the corresponding *eigenvector space.* Individual members of that space are called *eigenvectors.* Finding the eigenvalues, and then from these the eigenvectors is the problem at hand.

One approach to solving this problem is to find the eigenvalues first, and then from them, find the eigenvectors. To find the eigenvalues we use this theorem:

Theorem 5 *Let* $\mathbf{A} \in \mathcal{C}^{n\times n}$ *and* $\lambda \in \mathcal{C}$.

$$\lambda \text{is an eigenvalue of } \mathbf{A} \Longleftrightarrow \det(\lambda\mathbf{I} - \mathbf{A}) = 0$$

It turns out that if you expand that determinant, you obtain a polynomial

$$p(\lambda) = \det(\lambda\mathbf{I} - \mathbf{A}) = \sum_{k=1}^{n} a_k x^k$$

which is called the *characteristic polynomial*, so to find the eigenvalues we need find the roots of that polynomial. For matrices up to size $n = 4$ there are direct methods that will give you the solution. So for such matrices, the eigenvalue problem is trivial. For larger polynomials there are no direct methods, and in many cases there is no simpler way to solve polynomial equations except by reducing them to eigenvalue problems! In fact, it is quite common to solve even 3rd order and 4th order polynomials this way.

In Matlab we usually solve eigenvalue problems with the `eig` function. The call:

```
e = eig(A);
```

will return a vector containing the eigenvalues of `A`. The call:

```
[V,D] = eig(A);
```

produces a diagonal matrix `D` of eigenvalues and a full matrix `V` whose columns are the corresponding eigenvectors.

The grand strategy of all eigenvalue algorithms is based on the fact that the matrix $\mathbf{Z}^{-1}\mathbf{AZ}$ has the same eigenvalues as $\mathbf{A}$ for all non-singular $\mathbf{Z}$. What we want to do then is to apply the appropriate sequence of $\mathbf{Z}$ matrices that will diagonalize our matrix $\mathbf{A}$. Then, the diagonal elements of the diagonal matrix will also be it's eigenvalues. Of course, this is easier said than done. In what follows, we will sketch out how this is done, and introduce you to all the related nifty Matlab functions. You can use these functions to operate on a lower level, and experiment with the eigenvalue algorithms more directly.

5.1 Hessenberg Forms

The first thing we want to do to our matrix $\mathbf{A}$ is bring it to the so-called *Hessenberg Form.* The Hessenberg form of a matrix is zero bellow the first lower subdiagonal and non-zero everywhere else. The existence theorem for Hessenberg forms says that:

Theorem 6 *If* $\mathbf{A} \in \mathcal{C}^{n\times n}$ *then there exists a unitary* $\mathbf{P} \in \mathcal{C}^{n\times n}$ *such that*

$$\mathbf{P}^H\mathbf{AP} = \mathbf{H}$$

where $\mathbf{H}$ *is such that*

$$h_{ij} = 0, \forall(i,j) : i > j + 1$$

The theoretical emphasis here is that we can bring our $\mathbf{A}$ to this funky form by applying a sequence of *unitary* $\mathbf{Z}$ matrices. In your numerical analysis course you will learn that this guarantees the numerical stability of these repeatitive transformations.

In Matlab, you can get this far by applying the `hess` function:

```
H = hess(A);
```

will return the matrix $\mathbf{H}$. If you also want to look at $\mathbf{P}$, call:

```
[P,H] = hess(A);
```

5.2 Schur Decompositions

The next step is to converge to the *Schur decomposition.* Here are the relevant existence theorems:

Theorem 7 *If* $\mathbf{A} \in \mathcal{C}^{n\times n}$ *then there exists a unitary* $\mathbf{Q} \in \mathcal{C}^{n\times n}$ *such that*

$$\mathbf{Q}^H\mathbf{A}\mathbf{Q} = \mathbf{T} = \mathbf{D} + \mathbf{N}$$

where $\mathbf{D} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ *and* $\mathbf{N} \in \mathcal{C}^{n\times n}$ *is strictly upper triangular. Furthermore,* $\mathbf{Q}$ *can be chosen so that the eigenvalues* λ_i *appear in any order along the diagonal.*

This decomposition is called the *Complex Schur Form.* There is another one for real matrices:

Theorem 8 *If* $\mathbf{A} \in \mathcal{R}^{n\times n}$ *then there exists an orthogonal* $\mathbf{Q} \in \mathcal{R}^{n\times n}$ *such that*

$$\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \begin{pmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} & \cdots & \mathbf{T}_{1m} \\ 0 & \mathbf{T}_{22} & \cdots & \mathbf{T}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{T}_{mm} \end{pmatrix}$$

where each $\mathbf{T}_{ii}$ *is either a* 1×1 *matrix which is an eigenvalue of* $\mathbf{A}$ *or a* 2×2 *matrix whose eigenvalues are also eigenvalues of* $\mathbf{A}$.

This one is called the *Real Schur Form.*

Matlab provides with a function called `schur`. The call

```
[Q,T] = schur(A)
```

will return the real schur form if the matrix is real, and the complex schur form if the matrix is complex. If you don't want the real schur form, then the function `rsf2csf` will convert the inputted real schur form to complex:

```
[Q,T] = rsf2csf(Q,T);
```

How do we get to Schur from Hessenberg form? The simplest way to do it is with the so called QR algorithm:

1. First of all, we get the Hessenberg form of $\mathbf{A}$:

$$\mathbf{H}_0 = \mathbf{P}_0^T \mathbf{A} \mathbf{P}_0$$

2. Then we apply the following recurrence:

$$\begin{cases} \mathbf{H}_k = \mathbf{Q}_k \mathbf{R}_k & \text{QR decomposition} \\ \mathbf{H}_{k+1} = \mathbf{R}_k \mathbf{Q}_k \end{cases}$$

It can be proven that these repetive iterations are actually equivalent to applying unitary matrices for $\mathbf{Z}$ and that they will take the Hessenberg form all the way to *Real Schur form*! In your numerical analysis course you will learn more details about the various algorithms that take you from Hessenberg to Schur. With Matlab you can actually experiment with many of these algorithms and see how they perform.

5.3 Balancing

Before applying the QR algorithm it is a good idea to *balance* the matrix. By balancing we rescale the matrix by transforming it to $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$ where $\mathbf{D}$ is a *diagonal* matrix so that it it has approximately equal row and column norms. In matlab, you can balance a matrix with the `balance` function:

```
B = balance(A);
```

While experimenting with variations of the QR algorithm, you can see what happens as you include or not include balancing.

5.4 Miscellaneous

With symmetric matrices, the following miracles happen: the eigenvalues are real, the Hessenberg form is a tridiagonal matrix, and the QR algorithm doesn't merely take us to a real Schur decomposition, but it completely diagonalizes our matrix! An even more exciting development is that for square symmetric matrices the Schur decomposition and the SVD decomposition *are the same thing!* From this fact, plus a few theoretical results, one can obtain an algorithm for computing the SVD by reducing it to a symmetric eigenvalue problem. The details of this are described in *"Matrix Computations"* by Gene Golub.

Another application of eigenvalue problems is solving polynomial equations. As you may know, direct methods for finding polynomial roots, exist only for polynomials up to 4^{th} degree. Let

$$P(x) = \sum_{k=1}^{n-1} a_k x^k + x^n$$

be your polynomial. Form the matrix:

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{n-1} \end{pmatrix}$$

The eigenvalues of $\mathbf{C}$ are the roots of $P(x)$. $\mathbf{C}$ is called the *companion matrix* of the polynomial. In matlab, if an array `P` holds the coefficients of a polynomial with `p(1)` being the highest order coefficient, the companion matrix can be retrieved by the `compan` function.

```
A = compan(p);
```

Of course Matlab, in it's never-ending efforts to make this all easier for the user, provides you with a function called `roots` that you can use to do all this automatically. So, if you describe your polynomial with an array `p` such that

```
y = p(1)*x^d + p(2)*x^(d-1) + ... + p(d)*x + p(d+1)
```

then the roots can be obtained by simply saying:

```
r = roots(p);
```

You will obtain an array of roots in `r`. To verify your roots, you can use the `polyval` function to evaluate the polynomial. Just call:

```
y = polyval(p,x);
```

where `p` is the array with the polynomial and `x` is a scalar.

Finally, Matlab has commands that allow the user to solve more generalized eigenvalue problems. One such problem is the following. Given two square matrices $\mathbf{A}, \mathbf{B} \in \mathcal{C}^{n\times n}$ find $\lambda \in \mathcal{C}$ and $\mathbf{x} \in \mathcal{C}^{n\times 1}$ such that

$$\mathbf{Ax} = \lambda \mathbf{Bx}.$$

To solve this problem call:

```
[V,L] = eig(A,B);  % eigenvectors -> V, eigenvalues -> L
l = eig(A,B);      % eigenvalues  -> l
```

In the first version the eigenvectors are returned as columns of `V` and the eigenvalues as diagonals of `L`. In the second version an array of the eigenvalues is returned.

LINEAR SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS

ELEFTHERIOS GKIOULEKAS

The purpose of this handout is to summarize the theory of the matrix exponential. You may note that this is a different approach from the one proposed in your textbook. The main motivation for preferring the matrix exponential is that it provides a unified theory that can solve *all* problems, including all *forced* problems. For examples, see Bronson [1]. A detailed exposition of the theory is given by Apostol [2].

We want to solve a linear system of ordinary differential equations of the form

$$\frac{dx_a}{dt} = \sum_{b=1}^{n} A_{ab} x_b + f_b, \tag{1}$$

where f_b is a known forcing function, and $\mathbf{A} = [A_{ab}]$ is a matrix of constant coefficients. The essential result of the matrix exponentials theory is that the solution is given by

$$\mathbf{x}(t) = \exp(t\mathbf{A})\mathbf{x}(0) + \exp(t\mathbf{A}) \int_0^t \exp(-s\mathbf{A})\mathbf{f}(s), \tag{2}$$

where $\mathbf{x}(0)$ is the initial condition, which we know. The first term represents the homogeneous solution, whereas the second term represents the particular solution. The matrix exponential is given by

$$e^{\mathbf{A}} = \exp(\mathbf{A}) = \sum_{n=0}^{+\infty} \frac{\mathbf{A}^n}{n!}. \tag{3}$$

Note that this is a generalization of the Taylor series expansion of the standard exponential function. Naturally, $e^{\mathbf{A}}$ is a $n \times n$ matrix. The problem at hand is to evaluate the matrix exponential without having to evaluate an infinite sum, which is not practical.

1. Eigenvalues and eigenvectors

The evaluation of the matrix exponential requires the use of the concept of eigenvalues and eigenvectors. We say that a matrix $\mathbf{A}$ has eigenvalue λ with eigenvector $\mathbf{v}$, if the equation $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ is satisfied. To find the eigenvalues, you solve the following equation

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0. \tag{4}$$

Then, you substitute the numbers that you find to $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ and solve that to find the corresponding eigenvector.

The geometric interpretation is that the eigenvector represents a direction such that any vector that points towards that direction (or in the opposite direction) will not be *rotated* when you apply the matrix on it. It will only stretch or shrink, and the corresponding eigenvalue will tell you how much. If all the eigenvalues are distinct, then you're dealing with a very nice matrix, because you will also have corresponding eigenvectors that are orthogonal to each other. This means that every other vector can be decomposed as

$$\mathbf{x} = \sum_{a=1}^{n} c_a \mathbf{v}_a, \tag{5}$$

where the numbers c_a are the coordinates of the vector $\mathbf{x}$ using the eigenvectors $\mathbf{v}_a$ of the matrix $\mathbf{A}$ as a frame of reference. It follows that when you apply the matrix to an arbitrary vector, all that happens is that you're multiplying its coordinates with certain constant numbers, the eigenvalues λ_a , regardless of who this vector is. This only holds however *if you use the eigenvectors to define the coordinate system.* Furthermore, this may fail if you have repeated eigenvalues.

2. Matrix exponential

The most convenient technique for evaluating the matrix exponential was invented by Putzer. It is based on the following two claims. First, there will always exist numbers a_k such that the matrix exponential $e^{\mathbf{A}}$ can be written as a finite sum given by

$$e^{\mathbf{A}} = \sum_{k=0}^{n-1} a_k \mathbf{A}^k. \tag{6}$$

These numbers can be calculated by using the following claim: Suppose that we define a polynomial given by

$$p(x) = \sum_{k=0}^{n-1} a_k x^k. \tag{7}$$

Then, suppose that you know all the eigenvalues of the matrix $\mathbf{A}$. Then, for each eigenvalue λ you may obtain one equation as follows.

$$\begin{aligned} \lambda \text{ simple eigenvalue} &\Longrightarrow e^{\lambda} = p(\lambda) \\ \lambda \text{ double eigenvalue} &\Longrightarrow e^{\lambda} = p(\lambda) = p'(\lambda) \\ \lambda \text{ triple eigenvalue} &\Longrightarrow e^{\lambda} = p(\lambda) = p'(\lambda) = p''(\lambda) \end{aligned} \tag{8}$$

etc. Solving these equations together as a system, will give you the numbers a_k.

This is a straightforward calculation that always works. However, that doesn't mean that you will always have to do it. In certain cases, we can work it out in general and derive special results that will allow you to circumvent the need to solve the system of linear equations.

For example, if your matrix is nice enough to have n distinct eigenvalues, then the matrix exponential can be calculated from the following expressions.

$$\begin{aligned} \exp(t\mathbf{A}) &= \sum_{k=1}^{n} e^{\lambda_k t} \mathbf{L}_k \\ \mathbf{L}_k &= \prod_{j\in[n]-\{k\}} \frac{1}{\lambda_k - \lambda_j} (\mathbf{A} - \lambda_j \mathbf{I}) \end{aligned} \tag{9}$$

This result by itself covers most circumstances.

Most useful is the 2×2 case. In this case, we distinguish between two possibilities: that the eigenvalues are distinct, or that they are equal. Suppose that you have found the eigenvalues, and they are λ_1, λ_2. Then, if $\lambda_1 = \lambda_2 = \lambda$ then

$$\exp(t\mathbf{A}) = e^{\lambda t}(\mathbf{I} + t(\mathbf{A} - \lambda\mathbf{I})) \tag{10}$$

If $\lambda_1 \neq \lambda_2$ then

$$\begin{aligned} \exp(t\mathbf{A}) &= \frac{\mathbf{A} - \lambda_2\mathbf{I}}{\lambda_1 - \lambda_2} e^{\lambda_1 t} + \frac{\mathbf{A} - \lambda_1\mathbf{I}}{\lambda_2 - \lambda_1} e^{\lambda_2 t} \\ &= \frac{\lambda_1 e^{\lambda_2 t} - \lambda_2 e^{\lambda_1 t}}{\lambda_1 - \lambda_2}\mathbf{I} + \frac{e^{\lambda_1 t} - e^{\lambda_2 t}}{\lambda_1 - \lambda_2}\mathbf{A} \end{aligned} \tag{11}$$

This result is sufficient to solve all 2×2 problems. Keep in mind, that when you know the matrix exponential, that allows you to write down solutions for any kind of forcing functions.

3. The eigenvector method

Now let us consider the method used by the textbook. This method is expedient only when all the eigenvalues are distinct *and* you do not have forcing in your problem. It is also conceptually interesting.

Suppose that $\mathbf{v}$ is an eigenvector of the matrix $\mathbf{A}$ with eigenvalue λ. What happens if you apply the matrix $e^{t\mathbf{A}}$ on the eigenvector $\mathbf{v}$ instead? Using the definition of the matrix exponential, you may show that

$$\exp(t\mathbf{A})\mathbf{v} = e^{\lambda t}\mathbf{v}. \tag{12}$$

Suppose that you want to solve the unforced problem

$$\frac{dx_a}{dt} = \sum_{b=1}^{n} A_{ab}x_b. \tag{13}$$

The key is to write the initial condition as a linear combination of the eigenvectors

$$\mathbf{x}(0) = \sum_{a=1}^{n} c_a \mathbf{v}_a. \tag{14}$$

Using this, we may calculate the solution as follows:

$$\begin{aligned} \mathbf{x}(t) = \exp(t\mathbf{A})\mathbf{x}(0) &= \exp(t\mathbf{A}) \sum_{a=1}^{n} c_a \mathbf{v}_a \\ &= \sum_{a=1}^{n} c_a \exp(t\mathbf{A})\mathbf{v}_a = \sum_{a=1}^{n} c_a \exp(\lambda_a t)\mathbf{v}_a \end{aligned} \tag{15}$$

What you get is the method of the textbook. Did you see how the matrix exponential disappeared? The idea is that *if* you don't have forcing, and *if* the eigenvalues *are* distinct, then you can circumvent the computation of the matrix exponential by evaluating the *eigenvectors* instead, which is an easier computation.

Don't forget that it is easy to diagnose whether your solutions are correct by substituting them back to the original equations.

References

[1] R. Bronson. *Differential Equations.* Scaum Outline Series McGrew–Hill, New York, 1994.

[2] T. M. Apostol. *Calculus.* John Wiley & Sons, New York, 1969.

References

The following references were consulted during the preparation of these lecture notes.

(1) A. Pistofides (1992), "Algebra IV", unpublished lecture notes.
(2) K. Gkatzoulis and M. Karamavrou (1988), "Linear Algebra", Ekdoseis ZHTH.
(3) T.M. Apostol (1969), "Calculus, Vol. 2", Wiley.
(4) J. Aldous and R.J. Wilson (2004), "Graphs and Applications", Springer.
(5) J.A. Bondy and U.S.R. Murty (1976), "Graph Theory with Applications", Elsevier Science Publishing.

Lecture notes by Pistofides are available for download at

http://www.math.utpa.edu/lf/OGS/pistofides.html