Mathematical Methods for Physicsts

릴리스 0.0.1

Hyun Seong, Kim

2022년 01월 03일

차례

물리학에서 쓰이는 여러 수학적 기법들에 관한 개략적인 소개를 위한 책입니다.

제 1 장

기초 논리학과 증명

이 단원에서는 기초적인 논리학과 증명에 대해 배운다. 수학을 공부함에 있어서 어떻게 정확히 타당한 수학적 논증(증명)을 만들어 내는지 이해하고 있어야한다. 이를 위해서는 논증에 사용되는 논리 규칙들과 명제들에 대한 공부가 먼저 선행되어야 한다. 논리 규칙들은 수학적 문장들의 의미를 좀 더 명료하게 하고, 이들에 대한 이해를 높여준다.

이 단원에서는 이를 위해 기초적인 명제/술어 논리학과 증명에 관해 다룰 것이다.

1.1 명제와 술어 논리(Proposition and Predicate Logic)

1.1.1 명제(Proposition)

명제 (proposition)는 참/거짓을 **판정** 할 수 있는 문장을 의미한다. 즉, 명제인 문장은 참이나 거짓 둘중 하나의 값을 가지게 된다. 이때, 둘 모두일 수는 없는데, 이러한 경우 판정할 수 없는 문장으로 명제가 아니게된다. 명제를 나타내는 변수를 **명제 변수** (proposition variable)라 부른다. 명제 변수는 일반적으로 소문자 ($p, q, r, s \dots$)를 사용한다.

참/거짓을 진리값이라 하는 데, 일반적으로 참 (Truth)을 T , 거짓 (False)을 F 로 표기한다.

여러 명제들은 더 작은 명제들로 표현 될 수도 있다. 그러나, 더 작은 명제로 표현될 수 없는 명제도 존재하는데, 이를 **원자 명제** (atomic proposition)라 한다. 이러한 명제들로 부터 새로운 명제들을 파생시킬수 있는데, 이렇게 만들어진 명제를 **복합 명제** (compound proposition), 기존 명제에 작용해 새로운 복합 명제를 만들어내는 것을 **논리 연산자** (logic operator)라 한다.

논리 연산자는 기존 명제에 새로운 의미를 연결한다고 해서 논리적 연결사(logical connective)라 표현을 쓰기도 한다. 이러한 명제들을 다루는 논리를 **명제 논리** (Propositional logic)이라 부른다.

논리 연산자를 소개하기에 앞서서 **진리표** (Truth table)를 소개하고 가자. 진리표는 어느 명제가 가질수 있는 진리 값을 모두 나열한 것으로 첫 행에는 표기할 명제들을, 아래 행에는 각 열의 첫번째 명제가 가질 수 있는 진리 값을 기술한다. 독립적인 명제들을 왼쪽에 오른쪽에 해당 명제들로 만들어진 복합 명제를 기술한다.

예를 들어서 두 개의 명제 p, q 와 이 둘로 이루어진 복합 명제 r 이 있다면, 다음과 같이 쓸 수 있다.

표 1: 진리표 예시

p	q	r
T	T	
T	F	
F	T	
F	F	

r 이 있는 열의 각 행에는 p, q 의 진리값에 따른 r 의 진리값이 들어간다. 어느 한 복합 명제 r 이 $p_1, p_2, p_3, \ldots p_n$ 의 독립 명제들로 이루어져 있다면, 진리표는 총 $2^n + 1$ 개의 행을 가지게 된다[2] . 따라서 진리표는 적은 독립 명제들을 가진 복합 명제의 판정에는 사용가능하지만, 많은 명제들을 가지고 있는 복합 명제의 판정에는 비효율적이다.

1.1.2 논리 연산자(Logic operator)

기초적인 논리 연산자에 **부정** (negation), **연언** (conjunction) 그리고 **선언** (disjunction)이 있다.

Definition 1 (부정 (Negation)) 명제 p 에 대해, " p 의 부정"은 $\neg p$ 로 표기하며, " p 가 아닌 경우"를 의미한다.

p	$\neg p$
T	F
F	T

부정 기호는 매우 다양하다. 정의에 쓰인 $\neg p$ 뿐만 아니라, $\sim p, !p, p', Np, -p, \bar{p}$ 들도 쓰인다. 단순히 not p 로 표기하기도 한다. $\bar{p}$ 는 집합 기호에서 여집합이나 켤레 복소수의 표기에도 쓰이는 기호라 혼동의 우려가 있어 사용하지 않았다. $!p$ 는 프로그래밍 언어등에서 부정을 나타내는 데 흔히 쓰인다. 일반적으로 NOT 연산자라 부른다.

Definition 2 (연언 (Conjuntion)) 명제 p, q 에 대해, " p, q 의 연언"은 $p \wedge q$ 로 표기하며, " p 가 참이며, q 도 참이다."를 의미한다.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

단순히, p and q 나 $p\&q$ 로 표기하기도 한다. 일반적으로 AND 연산자라 부른다.

Definition 3 (선언 (disjuntion)) 명제 p, q 에 대해, " p, q 의 선언"은 $p \vee q$ 로 표기하며, " p 가 참이거나 q 가 참이다."를 의미한다.

p	q	$p \vee q$
T	T	T
T	F	T
F	F	T
F	F	F

[2] 열은 최소 $n + 1$ 에서 더 많아질 수도 있다. 복잡한 복합 명제는 하위 복합 명제를 중간열에 추가하기도 하기 때문이다.

단순히 p or q 로 표기하기도 한다. 일반적으로 "또는" 혹은 "or"로 번역하는 데, 논리학에서의 선언을 의미하는 "또는"과 "or"은 일상 언어에서의 의미와 차이가 있다. 일상 언어에서 해당 단어들은 **베타적** (exclusive) 의미를 가지고 있다.

예를 들어 "그 서버는 Windows이거나 Linux이다. "에서 "서버"는 Windows이면서 Linux일 수는 없다. 오직 한가지 경우만이 가능하다. 반면에, 위의 선언(disjunction)은 진리표에서 보이듯이 주어진 두 명제가 모두 참인 경우가 가능하다. 이러한 성질을 **포괄적** (inclusive)이라 부른다. 별도의 서술이 없을 경우 논리학에서 선언은 모두 포괄적으로 다루어진다. 베타적 의미의 선언은 따로 정의되어 있다. 일반적으로 OR 연산자라 부른다.

Definition 4 (베타적 선언 (Exclusive disjunction)) 명제 p, q 에 대해, " p, q 의 베타적 선언"은 $p \oplus q$ 로 표기하며, " p 나 q 둘 중 하나만이 참이다."를 의미한다.

p	q	$p \oplus q$
T	T	F
T	F	T
F	F	T
F	F	F

$p \veebar q, p + q, p$ xor q 로 쓰기도 한다. 별도의 베타적 선언 기호를 쓰지않고 포괄적 선언으로 표기할 수도 있다. 진리표를 기준으로 $p \oplus q$ 는 $(p \vee q) \wedge \neg(p \wedge q)$ 와 같다. 일반적으로 XOR 연산자라 부른다.

조건문(Conditional Statement)

위의 기본 연산자들 외에 **조건문** 이라는 방법으로 명제들을 결합할 수도 있다.

Definition 5 (조건문 (Conditional statement)) 명제 p, q 에 대해, 조건문 " $p \to q$ "는 "만약, p 이면, q 이다."를 의미한다. $p \to q$ 는 p 가 참인데, q 가 거짓일 경우에 거짓이고, 나머지 경우에는 모두 참이다. $p \to q$ 에서 p 는 전제(premise)라 하며, q 는 결론(conclusion)이라 한다.

p	q	$p \to q$
T	T	T
T	F	F
F	F	T
F	F	T

$p \to q$ 가 조건문이라 불리는 이유는 q 의 진리값이 p 의 진리값에 의존하기 때문이다. 이 조건문을 나타내는 표현도 매우 다양한데 다음이 있다.

$p \to q$		
if p, then q	if p , q	p implies q
p is sufficient for q	a necessary condtion for p is q	q when p
q if p	q whenever p	q unless $\neg p$
a sufficient condition for q is p	q follows from p	p only if q
p only if q	q is necessary for p	q provided that p

" p only if q "란 표현이 조금 혼동이 올 수도 있는데, " $p \to q$ "가 참임을 기본 전제로 두자, 이때, q 가 참이 아니면, p 도 절대로 참일 수 없다. p 가 참이기 위해서는 q 가 반드시 참이여만 한다. 때때로 " p only if q "를 $q \to p$ 로 받아들이는 경우도 있는 데, 완전히 반대로 해석한것이다. 자주 발생하는 실수이니 주의해야 한다. 많은 경우 수학적 정리와 문제들은 명제 기호로 쓰여있지 않다. 때문에 이를 해석할 때, 주어진 규칙에 맞게 해석했는지 유의해야 한다.

조건문과 같은 진리표를 가지는 복합 명제도 있는데, $p \to q$ 는 $\neg(p \wedge \neg q)$ 와 같은 진리 값을 가진다.

p	q	$p \to q$	$\neg(p \wedge \neg q)$
T	T	T	T
T	F	F	F
F	F	T	T
F	F	T	T

주어진 조건문 $p \to q$ 에 기반해, 명제의 순서를 바꾸거나 부정들을 이용해 새로운 명제들을 만들 수 있다. 이들을 각각 **역** (converse), **대우** (contrapositive), 그리고 **이** (inverse)라 부른다.

명제(propostion)	역(converse)	대우(contrapositive)	이(inverse)
$p \to q$	$q \to p$	$\neg q \to \neg p$	$\neg p \to \neg q$

이 중 대우는 본래 명제와 같은 진리값을 가진다. 대우가 참이면 본 명제도 참이고, 대우가 거짓이면 본 명제도 거짓이다. 역이나 이는 이런 성질을 가지지 않는다. 일반적으로 역이나 이의 진리값이 정해진다고 해도, 이를 통해 본래 명제의 진리값을 판정할 수는 없다.

다음의 진리 표를 참고하자.

p	q	$p \to q$	$\neg q \to \neg p$	$q \to p$	$\neg p \to \neg q$
T	T	T	T	T	T
T	F	F	F	T	T
F	T	T	T	F	F
F	F	T	T	T	T

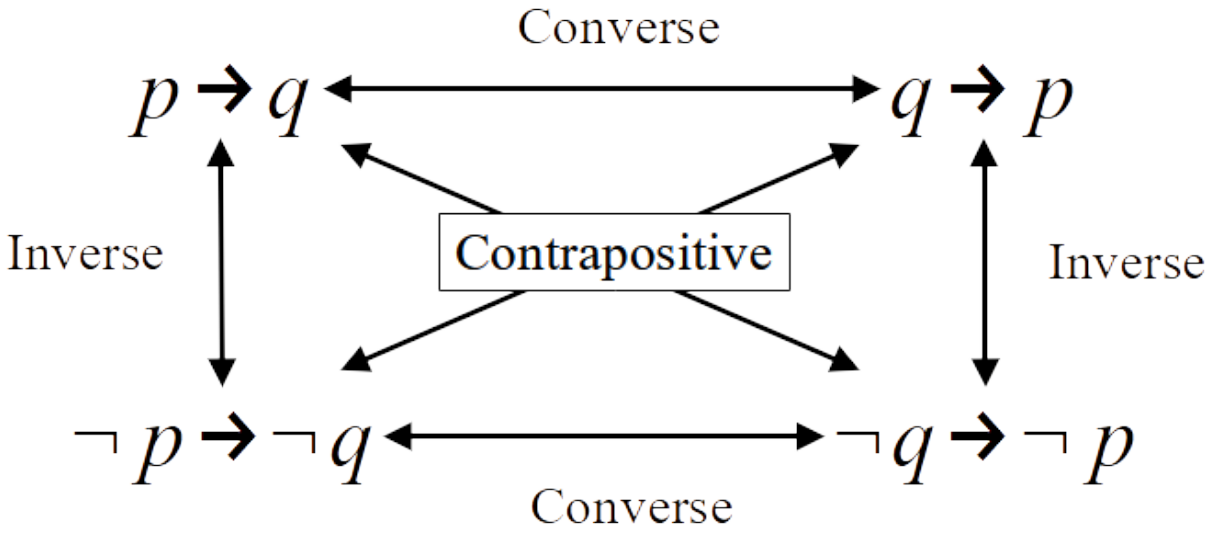


그림 1: 명제, 역, 대우의 구조 다이어그램

다이어그램에서 보이다 시피, 대우는 조건문에 역과 이를 가한 것과 같다. 이때 순서는 상관없다.

조건문과 조건문의 대우는 서로 동일한 진리값을 공유하는 데, 이는 조건문을 이루는 두 명제 p 와 q 의 진리값에 관계없이 가지는 성질이다. 이렇듯 어떤 두 복합 명제가 구성하는 독립 명제들의 진리값과 상관없이 언제나

동일한 진리값을 가질 경우 이 두 명제가 **논리적 동치** (Logical equivalence)에 있다 한다. 조건문과 그 대우, 조건문의 역과 조건문의 이는 각각 논리적 동치 관계에 있다.

어느 두 명제가 진리값을 공유함을 나타내는 복합 명제는 **쌍조건문** (biconditional statement)이라 한다.

Definition 6 (쌍조건문 (Biconditional statement)) 명제 p, q 에 대해, 쌍조건문 $p \leftrightarrow q$ 는 $(p \to q) \wedge (q \to p)$ 을 의미한다. 쌍조건문은 두 명제 p 와 q 가 같은 진리값을 가질 때 참을, 다른 경우에 거짓이다.

p	q	$p \to q$
T	T	T
T	F	F
F	F	T
F	F	T

이 쌍조건 관계에 있는 두 명제는 한가지 명제의 진리값으로 다른 명제의 진리값을 완전히 결정할 수 있다. 쌍조건문은 두 조건문 $p \to q$ 와 $q \to p$ 가 모두 참일때 성립한다. 이 둘을 나타내는 표현으로 각각 " p only if q ", " p if q "이 있는데, 일반적으로 이 둘을 같이 써서 " p if and only if q "라 한다. 이외에도 다음의 표현들이 있다.

$p \leftrightarrow q$	
p is necessary and sufficient for q	if p then q and conversely
p iff q	p exactly when q

1.1.3 논리 연산의 적용 순서(Precedence of Logical Operators)

복합 명제를 작성하게 될 때, 위에 서술한 부정, 선언, 연연 그리고 조건문(쌍조건문)들을 다양하게 조합해서 사용하게 된다. 이때, 해당 연산자들의 적용 범위와 순서에 대한 규약이 필요하다. 이러한 규정이 없는 경우 한 복합 명제가 적용 순서와 범위에 따라 다른 명제가 될 수 있기 때문이다.

예로 다음을 보자.

$$p \vee q \wedge \neg r$$

괄호(parentheses)로 연산 순서를 정해 묶어 보면, $(p \vee q) \wedge \neg r$ 이나 $p \vee (q \wedge \neg r)$ 가 가능하다. 위의 복합 명제는 이 두 명제중 어느 명제를 의미하는가? 연산의 순서가 정해지지 않을 경우 이러한 구분이 불가능하다. 부정, 선언, 연언, 조건 그리고 쌍조건 5가지의 연산자는 다음과 같은 우선순위를 가진다.

Operator	Order
$\neg$	1
$\wedge$	2
$\vee$	3
$\to$	4
$\leftrightarrow$	5

따라서 예시로 나온 $p \vee q \wedge \neg r$ 는 $p \vee (q \wedge \neg r)$ 을 의미한다. 적용 순서에 더해, $\neg$ 는 뒷 복합 명제 전체나 복합명제를 이루는 단일 명제에 적용 될 수도 있다. 괄호로 구분 되어있지 않을 경우 $\neg$ 는 가장 짧은 명제에 적용된다.

즉, $\neg r \wedge p \to q$ 는 $(\neg r) \wedge p \to q$ 를 의미한다. 전체 복합명제의 부정을 취하고 싶다면, 괄호를 사용해 $\neg(r \wedge p \to q)$ 로 표기해야 한다.

1.1.4 복합 명제의 성질 (Properties of Compound Proposition)

복합 명제는 명제들의 결합 구조에 따라 여러가지 성질을 가지게 된다. 특히 진리값에 대해, 합성된 복합 명제는 이루는 명제들의 진리값에 보편적으로 의존하지만, 특정 명제들은 구성 명제들의 진리값에 관계 없이 일관된 진리값을 유지할 수도 있다. 이러한 성질을 가지는 복합 명제를 **항진 명제** 와 **모순 명제** 라 부른다. 영어로 각각 **tautology** , **contradiction** 이라 한다.

복합 명제 중 항진, 모순도 아닌 명제를 **contingency** 하다라 한다.

논리적 동치 (Logical equivalence)

Definition 7 (논리적 동치(Logical Equivalence)) 복합 명제 p 와 q 에 대해, 명제 $p \leftrightarrow q$ 가 항진 명제(tautology)일 때, 이 두 명제가 **논리적 동치** (Logical equivalence) 관계에 있다라 하고, 이를 $p \equiv q$ 로 표기한다.

$\equiv$ 외에 $\Leftrightarrow$ 표기도 사용한다. 유의점은 논리적 동치는 논리적 연결사, 연산자가 아니다. 이는 단지 명제들 사이의 논리적 관계를 나타낼 뿐이고 단순히, " $p \to q$ 는 항진 명제이다"를 의미한다. 동치 관계에 있는 명제의 파악과 구성은 여러 명제의 진리값을 판별하는 데 매우 중요하다. 특정 명제의 진리값 판별이 매우 어려운 경우라도, 이와 동치 관계인 명제는 손쉽게 판정이 가능할 수도 있기 때문이다.

빈번히 쓰이는 논리적 동치 관계중 하나로 드모르간의 법칙(De Morgan's law)이 있다.

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$
$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

이 법칙은 확장해 n 개의 명제 $\{p_i\}_{i=1}^n$ 에 대해 다음과 같이 쓸 수 있다.

$$\neg(p_1 \vee p_2 \vee \dots p_n) \equiv (\neg p_1 \wedge \neg p_2 \wedge \cdots \wedge \neg p_n)$$
$$\neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \equiv (\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n)$$

줄여쓰면, $\neg(\bigvee_{i=1}^n p_i) \equiv \bigwedge_{i=1}^n (\neg p_i)$, $\neg(\bigwedge_{i=1}^n p_i) \equiv \bigvee_{i=1}^n \neg p_i$ 로 쓸 수 있다. 이 확장된 드모르간의 법칙의 증명은 수학적 귀납법을 사용해야 한다. 귀납법의 증명 이후 단원에서 자세한 내용을 보도록 하자.

혼동하지 말아야하는 점이 이후 대수 관계에서 나오는 동치 관계(equivalent relation)와 동치류(equivalence class)이다. 여기서 서술하는 논리적 동치와 동치 관계는 적용 대상이 다르다. 논리적 동치 관계는 명제들끼리 이루는 논리적 관계를, 동치 관계는 집합속 원소들 사이에 정의된 관계의 성질을 의미한다. 이 동치 관계는 논리학에서 동일성 관계라 부른다. 이 관계도 논리적 동치 관계에 속한다.

자세한 사항은 집합과 대수 공간을 볼 때 다루도록 하자.

충족성(statifiablility)

복합 명제에 대해, 복합 명제가 참인 진리값을 가지게 만드는 개별 명제들의 진리값 조합이 존재한다면, 이를 **충족하다** (satisfiable)라 한다. 이러한 진리값 조합이 존재하지 않는다면, 이를 **불충족하다** (unsatisfiable)라 한다. 충족성을 가지는 명제는 모순명제가 아니다. 즉, 항진 명제(tautology)나 contingency인 명제이다. 불충족한 명제는 해당 명제의 부정이 항진 명제인 것과 논리적 동치 관계를 이룬다.

어느 복합 명제가 충족성을 만족하느냐 만족하지 않느냐를 판정하는 문제는 복합 명제를 이루는 구성 명제의 수가 많을 수록 어려워진다. 적은 숫자의 경우는 단순히 진리표를 사용해서 보일 수 있지만, n 개의 명제로 이루어진 복합 명제는 2^n 의 경우의 수가 존재하게 된다. 이를 판정할 수 있는 효율적인 기계적 절차는 존재하지 않는다. 다만, 특정 범주의 명제들에 대해 실용적으로 판정 가능한 방법들이 연구되었다.

1.1.5 술어 (Predicate)

명제만으로 수학의 모든 문장을 표현할 수 있다면 좋겠지만, 실제로는 불가능하다. 대표적으로 다음과 같은 문장들을 생각해보자.

- 자연수 x 가 $x = 2$ 이다.
- 유리수 집합에 $x^2 = 3$ 을 만족하는 어떤 유리수 x 가 존재한다.
- 모든 실수 x 에 대해, 정수 n 이 존재해, $n < x < n + 1$ 을 만족한다.

윗 문장들의 진리값에 대해 생각해보자 이들은 x 가 실제로 어느 대상이냐에 따라 문장의 진리값이 달라진다. "자연수 x 가 $x = 2$ 이다."는 x 가 2 일 경우 참이지만, x 가 2 가 아닌 다른 자연수일 경우 거짓이 된다. 윗 문장들은 어느 특정한 대상이 아니라 특정한 범주에 대한 서술이다.

이들은 명제 논리의 언어로 표현이 불가능하다. 명제에서는 특정한 범주에 속한 대상을 지칭하는 언어가 없기 때문이다. 명제 논리를 확장한 더 보편적인 논리 체계를 필요로 한다. 이 논리 체계는 명제 논리를 그대로 포함하며 범주와 그 논리적 관계를 서술할 수 있는 체계가 될 것이다.

윗 문장들을 각각 두 부분으로 나누면, 변수: x 와 문장 속 변수의 성질을 서술하는 부분 문장; **술어** (predicate)로 나눌수 있다. 이 술어는 **명제 함수** (propositional function)이라 부른다. 변수 x 를 받아 진리값을 가지는 명제를 만들어내기 때문이다. 일반적으로 대문자로 $P, Q, \dots$ 로 표기하며 변수 x 에 대해, $P(x)$ 로 쓴다. 여러 변수가 함께 사용될 경우 $P(x, y, z, \dots)$ 로 쓴다.

어느 명제 p 는 특정한 값 a 와 술어 P 로 표현될 수 있다.

$$p = P(a)$$

어느 한 술어에 대해, 술어에 필요한 변수의 갯수로 술어를 구분짓기도 한다. 예를 들어서 다음 두 문장을 보자

x 는 자연수이다.

p 는 q 와 동치이다.

이 문장들은 각각 술어 P = "___ 는 자연수이다" , Q = "___ 는 ____ 와 동치이다" 와 독립 변수 (x), (p , q)로 이루어져 있다. P 와 같이 1 개의 독립 변수를 필요로 하는 술어를 **일항 술어** Q 와 같이 2 개의 독립 변수를 필요로 하는 술어를 **이항 술어** 라 부른다. $R(x_1, x_2, x_3, \dots, x_n)$ 와 같이 n 개의 독립 변수 $\{x_i\}_{i=1}^{n}$ 를 필요로 하는 술어는 n **항 술어** 이다.

술어는 명제의 구조를 분해해 변수와 그에 대한 서술로 이들을 나눈것이다. 이때, 변수가 가질 수 있는 값의 범주에 따라 술어의 진리값은 달라질 수 있다[1] . 이 때, 여러가지 상황이 가능하다. 어느 술어 $P(x)$ 에 대해,

- 1. x 범주에서 $P(x)$ 가 참인 x 는 존재하지 않는다.
- 2. x 범주에서 $P(x)$ 가 참인 x 는 1개 존재한다.
- 3. x 범주에서 $P(x)$ 가 참인 x 는 2개 존재한다. ⋮
- **n+1.** x **범주에서** $P(x)$ **가 참인** x **는** n **개 존재한다.** ⋮
- n+2. x 범주에서 모든 x 가 $P(x)$ 가 참임을 만족한다.

x 범주를 술어 P 로 정의할 수도 있다. 이 때는 정의에 따라 n+2을 자동으로 만족한다.

[1] 해당 집합에 대해, 논리학에서는 가능세계라는 표현을 쓰기도 한다.

양화사(Quantifier)

보편 양화사 (Universal quantifier)

존재 양화사 (Existential quantifier)

양화사의 부정, 결합, 분배 규칙

<!- 논리적 추론과 증명: 196-202 참고 ->

$$\neg(\exists x)Px \rightarrow (\forall x)\neg Px$$

$$\neg(\forall x)Px \rightarrow (\exists x)\neg Px$$

1.1.6 추론 규칙 (Rules of Inference)

기본 규칙

- 조건문 제거
- 선언 제거
- 선언 도입
- 연언 제거
- 연언 도입
- 쌍조건 도입
- 쌍조건 제거
- 부정 제거
- 조건 도입
- 부정 도입

파생 규칙

양화사 규칙

- ∀ 제거
- ∀ 도입
- ∃ 제거
- ∃ 도입

이 중 기본 규칙과 파생 규칙은 명제 논리의 추론 규칙이다. 술어 논리는 이에 더해 양화사 규칙을 추가로 포함한다.

보편 양화사의 도입: 임의의 대상 u 에 대해, A(u) 이다. -> $\forall x A(x)$

1.1.7 1 차와 2 차 논리 (First and Second order logic)

1차 논리는 원소에만 양화사를 가할 수 있고, 술어에는 한정 기호를 가할 수 없는 술어 논리를 의미한다. 본 문서에서 서술하는 술어 논리는 1차 논리를 의미하며 일반적으로 언급이 없을 경우 대부분의 술어논리는 1차 술어 논리이다. 2차 논리는 변수들의 집합, 술어에도 양화사가 적용가능하다.

이를 소개하는 이유는 특정 대상의 공리를 다지는 상황에서 공리(Axiom)뿐만 아니라 공리꼴(Axiom of schema)을 사용해서 정의하는 경우도 흔하기 때문이다. 공리는 어느 술어와 양화사가 적용되는 대상(원소)로 이루어져있는 1차 술어 논리지만, 공리꼴은 2 차 논리이므로 술어에도 양화사가 적용될 수 있다. 이를 이용해 임의의 성질을 만족하는 대상들의 존재성을 표현할 수 있다. ZFC 공리계에서 이러한 상황을 살펴볼 수 있다.

1.2 증명(Proof)

1.2.1 용어

공리(Axiom)는 참이라고 가정하는 명제를 의미한다. 공리계(Axiom system)는 이러한 공리들을 모아 놓은 것을 의미한다. 공리는 주어진 명제들에 대해, 참 거짓을 판정할 수 있는 전제로써 사용할 수 있다. 이때, 주어진 명제는 참/거짓으로 판정이 되거나 참/거짓 모두 불가능할 수도 있다. 참/거짓으로 판정된 명제는 정리(Theorem)라 불린다.

이러한 명제의 판정을 위한 논증을 증명(Proof)라 부른다. 증명의 과정에서 별개의 명제들을 판별해야 할 수도 있는데, 증명 과정에서 보조로 판정한 명제들을 보조 정리(Lemma)라 부른다. 어떤 명제가 주어진 공리계에서 참이라 판정되었다면, 다른 명제의 증명에서 공리계의 공리와 함께 증명 과정에서 사용가능하다.

별도의 추가적인 공리 없이 바로 증명된 정리에서 유도되는 정리가 있는데, 이는 따름정리(Corollary)라 부른다.

1.2.2 논증

1.2.3 직접증명(Direct proof)

논리적 동치 관계의 연쇄

논리적 동치 관계는 transitivity하다. $p \leftrightarrow q$, $q \leftrightarrow r$ 로 부터 $p \leftrightarrow r$ 을 유도해낼 수 있다.

1. $p \leftrightarrow q$
2. $q \leftrightarrow r$
3. $p \rightarrow q$
4. $p \rightarrow q$

1.2.4 간접증명(Indirect proof)

대우증명(Proof by contraposition)

모순증명(Proof by contradiction)

1.2.5 다른 증명 방법들

사례별증명(Exhaustive proof/Proof by cases)

진리표를 사용한 증명과 정확히 일치한다. 특정한 경우, 증명해야 하는 복합 명제의 경우가 유한하거나 무한하지만 범주를 나누어 무한한 경우에 대한 증명이 끝나고 유한한 경우만 남을 수도 있다. 이 경우 사례별로 증명을 할 수도 있다.

대표적인 사례가 4색 정리(Four color theorem)의 증명이다. 이 증명은 이러한 사례별 증명에서 사례를 컴퓨터 프로그램으로 증명한 경우이다.

WLOG : Without Loss Of Generality

존재성(Existence) 유일성(Uniqueness)

이곳에 서술한 증명이 모든 증명 전략은 아니다. 상황에 따라서 다양한 증명이 있을 수 있고, 각 수학 분야에서 쓰이는 또다른 증명법이 있기도 하며, 특정 문제에서 특이한 증명이 사용되기도 한다. 그러나 공통적으로 모두 논증의 타당성을 증명하는 과정일 뿐이다. 어느 단 하나의 올바른 증명이 있지는 않다. 증명의 각 단계가 논리적으로 옳다면 모두 올바른 증명이다. 위의 증명 방법들과 논리학에 대한 지식이 있다고 해서 증명을 잘할 수 있지는 않다. 증명에 있어 기계적 방법은 존재하지 않는다. 많은 연습을 통해서 익숙해지는 것밖에 없다.

수학적 귀납법(mathematical induction)은 적어도 고등학교 교육과정에서도 배우지만, 이 단원에서는 생략했다. 정확한 서술을 위해서는 정렬 가능성(order)와 자연수의 성질을 알고있어야하기 때문이다. 귀납법의 가장 큰 약점은 반례의 존재성을 제거할 수 없다는 점이다. 어떤 이론이 단순히 1 에서 1000 까지 아니면 계산할 수 있는 큰 수까지에서 참이라 하더라도 이는 특정 범주내에서 참임을 보일 뿐이지, 전체 수에 관해서는 무엇도 보장할 수 없다. 실제로 아주 큰 수에서 대소 관계의 역전이 일어나는 함수들도 존재한다[3] . 수학적 귀납법은 몇가지 공리를 이용해 자연수 범주내에서의 명제 진리값의 일관성을 유지시킬 수 있다.

1.3 귀납의 문제와 과학적 방법론

과학적 방법론은 귀납-연역적 방법을 모두 사용하는 방법이다. 한가지 문제는 우리가 자연에 대한 정보를 얻는 방법이 근본적으로 귀납적인 절차라는 점이고 우리가 세운 가설을 검증하는 방법 또한 귀납적인 절차라는 점이다. 때문에 실제 자연과학의 이론은 **입증** 될 수 없다는 표현을 쓰는 학자들도 많다. 오로지 **반증** 만이 가능하다. 이러한,

본질적으로, 반증되지 않는다면 참으로 가정한다.

[3] Skewes, Stanley. “On the Difference π (x)- lix (II).” Proceedings of the London Mathematical Society 3.1 (1955): 48-70.

1.4 참고문헌

- Rosen, K.H., Discrete Mathematics and Its Applications 8th Paperback, ISBN:9781260091991, 2018, McGraw-Hill Education.
- 이병덕, 논리적 추론과 증명 3rd, ISBN:9788956441290, 2015, 이제이북스.
- von Plato, Jan, "The Development of Proof Theory", The Stanford Encyclopedia of Philosophy (Winter 2018 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2018/entries/proof-theory-development/>
- Hammack, R.H, Book of Proof 3rd, ISBN:9780989472135, 2019, Richard Hammack
- Hepburn, Brian and Hanne Andersen, "Scientific Method", The Stanford Encyclopedia of Philosophy (Summer 2021 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/sum2021/entries/scientific-method/>.
- Judson, T.W, Abstract Algebra: Theory and Applications, ISBN:9781944325107, 2019, Orthogonal Publishing L3C.

제 2 장

집합과 대수 공간

이 단원에서는 기초적인 집합과 대수 구조에 관해 다룬다. 집합에 대한 소개와 여러 대수 구조들; 군, 환, 체 등에 대한 개략적인 소개를 한다. 해당 단원이 관련 개념들에 대해, 많은 내용을 포함하고 있지는 않다 전 단원과 같이 이 단원은 다른 수학 단원을 보조하기 위한 역할이다. 대수 구조들에 관한 자세한 내용은 차후 대수학 단원에서 다루고, 여기서 예시로 다루는 실수, 정수 등의 내용도 우리가 일반적으로 알고 있는 성질에 의존할 것이다. 해당 내용의 엄밀한 구축은 해석학 입문에서 다루도록 하자.

2.1 기초 집합론(Introduction to Set Theory)

집합이라는 대상은 수학적 공리로 정의하기에는 상당히 에매한 대상이다. 무엇보다 집합은 다른 수학적 대상을 정의하는 데 가장 기초적으로 사용되는 만큼 다른 정의에 의존할 수도 없다.

용어에 대해 자연어적인 정의를 우선하고 가도록 하자

2.1.1 소박한 집합론 (Naive Set Theory)

소박한 집합론은 이름 그대로 소박한(Naive)한 집합론이다. 소박하다라는 의미는 형식적(formal)한 정의가 아닌 비형식적인 자연언어로 정의되어 있다라는 것을 의미한다. 이는 처음으로 정의된 집합론으로 Georg Cantor가 무한 집합에 관해 연구할 때 도입하였다. 이러한 소박한 집합론에서 집합의 용어들은 다음과 같이 자연언어로 정의된다.

1. **집합(Set)** 은 잘 정의된 객체들의 모음이다(Well-defined collection of objects).
2. 집합이 포함하는 객체는 **원소(elements)** 라 부른다.
3. X 가 집합이고 a 가 집합 X 의 원소라면, X 가 a 를 포함한다, a 가 X 에 속한다라 한다. $z \in X$ 로 표기한다.
4. X 가 집합이고 a 가 집합 X 의 원소가 아니라면, $z \notin X$ 로 표기한다.
5. 두 집합 X , Y 에 대해, X 의 모든 원소들이 Y 의 원소이면, X 는 Y 의 부분 집합(Subset)이라 부르며, $X \subset Y$ 로 표기한다.
6. 집합의 원소들이 집합을 결정한다. 즉, 동일한 원소를 가지면서 다른 두 집합은 존재하지 않는다.
7. 임의의 성질에 대해, 이를 만족시키는 원소만을 원소로 가지는 집합이 존재한다.

1-5는 집합론의 용어를 정의하고, 6-7은 집합론의 공리이다. 소박한 집합론은 2개의 공리를 포함한다. 이 둘은 각각 외연 공리와 Unrestriced Comprehension 공리라 불린다. 이 둘을 다시 쓰면 다음과 같다.

Axiom 8 (외연 Extensionality)

$$\forall x \forall y [\forall z (z \in x \leftrightarrow z \in y) \rightarrow x = y]$$

Axiom Schema 9 (Unrestricted Comprehension) 집합론의 언어로 된 술어 $P(x, w_1, \ldots, w_n)$ 에 대해,

$$\forall w_1, \ldots, w_n \exists z [z \in x \leftrightarrow P(z, w_1, \ldots, w_n)]$$

일반적으로 부르는 집합은

존재 공리

2.1.2 러셀의 역설과 ZFC 공리계

러셀의 역설 (Russell's Paradox)

집합뿐만 아니라 술어로 가보자,

ZFC 공리계 (ZFC Axioms)

ZFC 공리는 체르멜로(**Z** ermelo)와 프렝켈(**F** raenkel)이 만든 공리계와 선택 공리(Axiom of **C** hoice)를 추가한 공리계를 의미한다.

Axiom 10 (확장 Extensionality) forall x forall y [forall z (z in x leftrightarrow z in y) rightarrow x =y]

정칙성 (Regularity)

Restricted comprehension

짝 (Pairing)

합 (Union)

치환 (Schema of Replacement)

무한 (Infinity)

벽 집합 (Power Set)

선택 (Choice)

선택 공리는 ZF 공리계와 함께 동치인 명제들을 찾을 수 있다.

- 초른의 보조정리(Zorn's lemma)
- 정렬 가능성(Well-Ordering Principle)
- 티호노프의 정리

이 외에도 동치인 정리들이 몇 있는데, 해당 개념이 필요할 때 설명하도록 하자.

2.1.3 정렬(Order)

2.1.4 자연수(Natural Number)

2.1.5 집합의 크기(Cardinality)

가산 집합과 불가산 집합(Countable and Uncountable Set)

Schröder-Bernstein theorem

두 무한 집합의 크기 비교에 관한 정리

2.2 대수구조(Algebraic Structure)

2.2.1 대수 구조(Algebraic Structure)

대수학에서 다루는 대상은 기본적으로 대수 구조와 변환이다. 여러 특정한 구조와 변환이 앞으로 등장하고, 각각 다양한 성질이 있지만 가장 먼저 해야할 일은 이 두 가지 대상이 무엇인지 알아보는 것이다. 대수 구조는 다음과 같이 정의된다.

Definition 11 (대수 구조 Algebraic Structure) 공집합이 아닌 집합 S 와 그 위에 정의된 연산 $\cdot$ 에 대해 $(S, \cdot)$ 을 **대수 구조** 라 한다.

집합 S 에 대해

$$\cdot : S \times S \to S$$

인 대응 관계를 **이항 연산(Binary operation)** 이라 부른다. 연산의 정의는 몇개의 원소를 포함하느냐에 따라 달라진다. n 개의 원소를 다른 원소에 대응 시키는 관계는 n 항 연산이라 부른다.

이러한 대수 구조는 집합과 그 위에 정의된 연산 각각에 따라 성질이 달라진다. 따라서 이를 표기에 반영해 다음과 같이 표기한다.

$$(S, \cdot)$$

$(S, +), (S, \times), (F, \times)$ 는 모두 각각 다른 대수 구조이다.

실수 전체 집합은 $\mathbb{R}$ 로 표기하는 데, 우린 이 집합의 원소(실수)의 연산(덧셈, 곱셈, 등호 그리고 부등호)을 익숙하게 하고 있다. 이 $\mathbb{R}$ 또한, 서술한 연산과 함께 대수 구조를 형성한다.

먼저, 각각의 연산자에 대해서 다음과 같이 표기 할 수 있다.

$$(\mathbb{R}, +), (\mathbb{R}, \times), (\mathbb{R}, <)$$

여러가지 연산이 정의된 경우 한꺼번에 표기 하기도 한다.

$$(\mathbb{R}, +, \times, <)$$

이처럼 대수 구조는 같은 집합이라 하더라도 연산자를 어떻게 정의하는 가에 따라 여러가지 구조를 만들 수 있다. 대수 구조의 종류로 대표적으로 다음이 존재한다.

1. 집합 (Set)
2. 모노이드 (Monoid)

3. 군 (Group)
4. 환 (Ring)
5. 체 (Field)

연산자 수	0	1	2
대수 구조	집합	모노이드, 군	환,체

이러한 구조 하나하나에 대해서 수학 분야 1개가 필요할 정도로(eg. 군론) 이들이 가지는 수학적 유용성과 가치는 매우 높다. 하지만 이 단원에서는 군, 환, 체의 개념과 기초적인 내용만 다루도록 하자. 이러한 분야에 대해 더 공부를 하고 싶다면 차후 현대 대수학 등의 강의를 들어 보는 것을 추천한다.

2.2.2 군(Group)

Definition 12 (군 Group) 대수 구조 $(S, \cdot)$ 가 다음 4가지 성질(군 공리계)을 만족할 때, 이를 **군** (Group)이라 한다.

1. **연산에 대해 닫힘 (Closure)**

$$\forall a, b \in S$$

$$a \cdot b, b \cdot a \in S$$

2. **항등원의 존재 (Existence of neutral/identity element)**

$$\exists 1 \in S \; and \; \forall \; s \in S$$

$$1 \cdot s = s \cdot 1 = s$$

3. **역원의 존재 (Existence of inverse element)**

$$\forall s \in S, \; \exists s' \in S$$

$$s \cdot s' = s' \cdot s = 1$$

$$s' = s^{-1} \;\; \text{inverse}$$

4. **결합 법칙(Assosiative law)**

$$\forall s_1, s_2, s_3 \in S$$

$$s_1 \cdot (s_2 \cdot s_3) = (s_1 \cdot s_2) \cdot s_3$$

5. **교환 법칙(Commutative law)**

$$\forall s_1, s_2 \in S$$

$$s_1 \cdot s_2 = s_2 \cdot s_1$$

5.를 추가로 만족하는 구조를 **가환군** 다른 말로 **아벨군** (Abelian Group) 이라 부른다[1] .

참고: 1,2,4 만을 만족 할 때, 이를 모노이드라 부른다.

초등학교에서 꾸준히 써왔던 사칙연산 $(+, -, \times, \div)$ 을 생각해보자, 실수는 덧셈 $+$ 과 곱셈 $\times$ 에 대해서 아벨군을 이룬다. $-$ 와 $\div$ 는 $+$,:math:times 에서 원소의 역원을 연산하는 것일 뿐, 별도로 정의된 연산자가 아니다[2] .

우리에게 익숙한 수 집합($\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$) 중 $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ 는 덧셈 $+$ 에 대해 군을 형성한다. 곱셈에 대해서도 $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ 는 군을 형성하지만, 정수 집합 $\mathbb{Z}$ 은 군을 형성하지 않는다.

어느 대수 구조가 군임을/군이 아님을 보인다는 것은 그 대수 구조가 군 공리계를 만족/불만족함을 보인다는 것이다. 다음을 통해 예를 들어보자

$$(\mathbb{Z}, +), (\mathbb{Z}, \times)$$

$(\mathbb{Z}, +)$

1. **연산에 대해 닫힘(Closure)**

$$\forall a, b \in \mathbb{Z}$$

$$c := a + b \text{ also } \in \mathbb{Z}$$

2. **항등원의 존재 (Existence of identity)**

$$\forall a \in \mathbb{Z}$$

$$0 \in \mathbb{Z} \text{ and } a + 0 = 0 + a = a$$

3. **역원의 존재 (Existence of inverse element)**

$$\forall a \in \mathbb{Z}$$

$$\exists - a \in \mathbb{Z} \text{ and } a + (-a) = (-a) + a = 0$$

4. 결합 법칙 (Associative law) .. math:

```
\forall a ,b,c \in \mathbb{Z}\\
(a+b) +c = a+(b+c)
```

따라서 $(\mathbb{Z}, +)$ 는 위의 4가지 아벨군 공리들을 만족하므로 아벨군을 이룬다.

$(\mathbb{Z}, \times)$

3.역원의 존재 .. math:

```
\forall a \in \mathbb{Z} \\

\nexists a^{-1} \in \mathbb{Z} \text{  such that } a\times(a^{-1}) = (a^{-1}) \times a=1
```

[1] 군론을 연구한 노르웨이 수학자 닐스 헨라크 아벨(Niels Henrik Abel)의 이름을 땄다. 이 책에서는 가환군 보다는 아벨군을 사용할 것이다.

[2] 물론 다른 형태로 이들을 연산으로 정의할 수도 있다. 다만, 역원 공리를 만족한다면 그 자체로 충분하다는 것이다.

이러한 역원이 존재하지 않으므로 $(\mathbb{Z}, \times)$ 는 군 공리들을 전부 만족시키지 못한다. 따라서 이는 군을 형성하지 않는다[3] (1,2,4 는 만족한다.)

2.2.3 체와 표수

체(Field),환(Ring)

Definition 13 (체 field) 대수 구조 $(F, +, \cdot)$ 가 다음을 만족할 때, 이를 **체** (Field)라 한다.

1. $(\mathbb{F}, +)$ 가 아벨군이다.
2. $(\mathbb{F} \setminus \{0\}, \cdot)$ 가 아벨군이다.
3. $+, \cdot$ 이 분배법칙(Distributive)을 만족한다.

$$(a+b) \cdot c = a \cdot c + b \cdot c \; \forall a, b, c \in \mathbb{F}$$

유의할 점이 미적분학을 배우고 왔다면, 스칼라나 벡터 장(field)에 대해서도 배웠을 터인데, 영어 단어가 똑같이 장과 체에 대해 field 로 쓰여, 같은 대상인지 혼동이 올 수도 있다. 정의에서 보다시피 다른 대상이다. 여기서 말하는 field 는 대수 구조의 하나를 말하는 것이고, 미적분에서 배운 field 는 수학적 공간에 스칼라, 벡터 아니면 텐서가 대응된 공간을 말하는 것이다.

이러한 혼란은 본래, 이 대수 구조가 처음 만들어졌을 당시, 독일어로 영역을 나타내는 Bereich 란 이름을 가지고 있었는 데, 영어권에서 해당 개념을 번역해 사용할 때, field 를 사용해 나타났다. 유럽권에서는 체를 나타내는 단어로 라틴어 corpus (영어의 body)를 어원으로 한 단어를 사용한다.

2번에서 $(\mathbb{F} \setminus 0, \cdot)$ 의 종류에 따라 체가 아닌 다른 대수 구조가 만들어진다.

- 아벨군: 체
- 군: 가환체(Skew-field)
- 모노이드: 가환환(Commutative-ring)
- 가환모노이드(Commutative-monoid): 환(Ring)

$(\mathbb{F}, +, \cdot)$ 에서 $+$ 를 덧셈, $\cdot$ 을 곱셈이라 한다.

체는 두 개의 연산자가 모두 아벨군임을 만족하고 그 사이의 분배 법칙이 성립할 때, 만들어진다.

군에서 이야기한 ($\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$)와 덧셈 $+$, 곱셈 $\times$ 연산자를 살펴보도록 하자. $\mathbb{N}$ 와 $\mathbb{Z}$ 는 $+$ 에 대해 아벨군을 이루지만, $\times$ 에 대해서는 아벨군을 이루지 않는다. 따라서 이들은 제외하고 남은 집합 중 $\mathbb{Q}$ 에 대해 생각해보자

$$(\mathbb{Q}, +, \times)$$

이 대수 구조는 체를 형성한다.

$(\mathbb{Q}, +, \times)$ 뿐만 아니라 $(\mathbb{R}, +, \times)$, $(\mathbb{C}, +, \times)$ 또한 대표적인 체이다.

다음은 체의 대표적인 성질이다.

Theorem 14

$$\forall a, b, c \in F, arbitrary$$
$$a + b = c + b \rightarrow a = c$$
$$if \; b \neq 0 \;\; a \cdot b = b \cdot c \rightarrow a = c$$

[3] 이러한 대수 구조가 공리들을 만족시키는 것을 명확히 보이기 위해서는 집합과 연산자를 정의한 이후에 그들의 정의와 성질로부터 공리계를 만족하는지 아닌 지를 판별해야 한다. 위의 내용은 단순히 우리가 익숙한 집합과 연산을 이용해 판별하는 예시를 보인 것으로 정확한 설명이 아니다. 일례로 단순히 정수의 집합을 정의부터 시작하면 상당히 긴 페이지를 할애해야 한다.

Corollary 15 체에서 각 연산의 항등원(0,1)은 유일하다.

우리가 실수 체 $\mathbb{R}$ 에서 익숙하게 쓰는 0,1의 성질들이 체 자체의 정의로부터 비롯된다.

Theorem 16 모든 체 F 와 $\forall a, b \in F$ 에 대해 다음이 성립한다.

$$a \cdot 0 = 0 \cdot a = 0$$
$$-(a \cdot b) = (-a) \cdot b = a \cdot (-b)$$

Corollary 17 0은 곱셈 역원이 없다.

표수(Charateristic)

Definition 18 (표수 Charateristic)

$$char(F) \in \mathbb{N}$$
$$char(F) := \begin{cases} min\{n \in \mathbb{N}_+ : \overbrace{1+1+1+\cdots+1}^{n} = 0\} \text{if n exist} \\ 0 \text{otherwise} \end{cases}$$

예를 들어 $char(\mathbb{Q}) = char(\mathbb{R}) = char(\mathbb{C}) = 0$ 이다.

Theorem 19 (Prime property of charateristic) $char(F) = n$ 에서 n 은 0 아니면 소수이다.

유한체(Finite Filed)

지금까지는 $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ 등의 원소 갯수가 무한한 체만을 다루었지만, 사실 유한한 원소로 이루어진 체도 존재한다. 대표적인 예시가

$$\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, \ldots, n-2, n-1\}$$

에서 $n = prime$ 인 경우이다[4] . 이때, .. math:

```
(\mathbb{Z}_n,+,\cdot)
```

은 체를 이룬다. 단, $n \neq prime$ 인 경우에는 해당하지 않는다. 이 경우 환을 이룬다.

Ordered Field

어느 Order 연산이 정의된 집합이 체 공리계를 만족한다면 이는 Ordered Field라 불린다. 이 Ordered Field는 특이한 성질이 있는데,

[4] $\mathbb{Z}_n$ 는 합동류(conguence class)로 번역된다. 각각의 원소 n 은 합동식의 결과가 같은 정수 전체를 의미한다. 예를 들자면 $\mathbb{Z}_5$ 에서 $7 \equiv 2 (\text{mod } 5)$ 이므로 $7 \equiv 12$mathbb{Z}_5` 의 어느 한 원소 3은 $n \equiv 3 (\text{mod } 5)$ 인 정수 전체를 의미하는 것이다.

2.3 참고문헌과 추가자료

- Judson, T.W, Abstract Algebra: Theory and Applications, ISBN:9781944325107, 2019, Orthogonal Publishing L3C.

제 3 장

실수와 복소수

정수(Integer)

유리수(Rational Number)

실수(Real Number)

무리수의 존재성(Existence of Irrational Number)

연산(Operation)

데디킨트 절단을 통한 실수 구축(Construction of Real number with Dedekind Cut)

확장된 실수(Extended Real Number)

연산 규칙과 원소를 추가해 주어야한다. 양의 무한대와 음의 무한대를 포함하는 집합 ${infty, -infty}$을 실수 $mathbb{R}$에 합해 다음을 얻는다.

$$widetilde{mathbb{R}} := mathbb{R} cup {infty, -infty}$$

이제 추가된 두 원소 $infty, -infty$와 $mathbb{R}$의 원소간의 연산 규칙을 정의해야 한다.

성질(Properties)

완비성(Completeness) #### 크기(Cardinality of Real number)

연속체 가설(Continuum Hypothesis)

복소수(Complex Number)

사원수(Hamilton Number)

제 4 장

벡터 공간

본 단원에서는 선형 대수학의 핵심 내용인 벡터 공간에 대해서 다룬다. 고등학교 시절과 대학교 1학년의 미적분학, 일반 물리학 등에서 벡터라는 개념에 대해 이미 공부를 했을 것이다. 이러한 과정에서 벡터는 **크기**와 **방향**을 가지는 양이라 정의한다. 수학에서 해당 과정에서 배우는 벡터는 유클리드 벡터를 의미한다.

선형 대수학에서 벡터의 정의는 사뭇 다르다. 여러 심화 수학 부분에서는 우리가 알고 있는 개념들이 다른 정의나 수학적 용어들을 사용해 재정의되는 경우가 많은데, 이러한 엄밀한 재정의는 대상의 수학적 모호성을 줄이고 개념의 오용을 방지한다[1] . 선형대수학 이후 수학에서 우리가 쓰는 벡터는 결국 **벡터 공간의 원소**를 의미할 것이다. 미적분학의 벡터는 특정한 벡터 공간의 원소이다. 이를 유클리드 공간이라 부른다.

4.1 벡터 공간 (Vector Space)

Definition 20 (벡터 공간 Vector Space) 어떤 체 $\mathbb{F}$ 와 집합 V , 그리고 이항 연산자

$$+ : V \times V \to V$$
$$\cdot : \mathbb{F} \times V \to V$$

가 다음의 공리 5개를 만족할 때, 이를 체 $\mathbb{F}$ 위의 **벡터 공간** 이라 부른다.

- $(V, +)$ 가 아벨군을 이룬다.
- $1 \cdot \mathbf{v} = \mathbf{v} \in V, 1 \in \mathbb{F}$
- $(\lambda_1 \lambda_2) \cdot \mathbf{v} = \lambda_1 \cdot (\lambda_2 \cdot \mathbf{v}) \forall \lambda_1, \lambda_2 \in \mathbb{F}, \mathbf{v} \in V$
- $\lambda(\mathbf{a}+\mathbf{b}) = \lambda\mathbf{a} + \lambda\mathbf{b} \forall \lambda \in \mathbb{F}, \mathbf{a},\mathbf{b} \in V$
- $(\lambda_1 + \lambda_2)\mathbf{a} = \lambda_1\mathbf{a} + \lambda_2\mathbf{a}$

이때, V 의 원소 $\mathbf{a} \in V$ 를 **벡터(vector)** 라고 부르며, $\mathbb{F}$ 의 원소 $\lambda \in \mathbb{F}$ 를 **스칼라(scalar)** 라 부른다.

$\mathbf{x},\mathbf{y} \in V, \lambda \in \mathbb{F}$ 에 대해, $\mathbf{x}+\mathbf{y}$ 를 벡터의 합(sum)이라 부르고, $\lambda \cdot \mathbf{x}$ 를 스칼라 곱(product)이라 부른다.

벡터 공간의 공리 1.을 보면 $(V, +)$ 가 아벨군을 이루는 것을 알 수 있다. 따라서 아벨군 공리계에 따라 이는 항등원(identity element)를 가짐을 알 수 있다. 이러한 항등원은 **영 벡터(0-vector)** 라 불린다.

영 벡터는 일반적으로 $\mathbf{0}$ 로 표기한다. 여러 공간을 동시에 다룰 때에는 어느 공간의 항등원인지 표기 하기 위해 $\mathbf{0}_V$ 등으로 표기하기도 한다.

이러한 벡터공간의 예시로 여러가지가 존재한다. 하나하나 살펴보면,

[1] 이러한 재정의가 이루어지지 않은 경우의 대표적인 예시로 극한이 있다. 엡실론-델타 논법이 나오기 전 극한의 활용은 상당히 무분별하게 적용되었다.

- 체와 체의 n 튜플 $(\mathbf{F}, \mathbf{F}^n)$
- 체위에 정의된 $m \times n$ 행렬 공간: $M_{m\times n}(\mathbb{F})$
- 공집합이 아닌 정의역에서 체로 정의된 연속 함수들
- 체 위에서 정의된 다항식
- 체로 정의된 수열

이 외에도 많은 벡터 공간들이 존재한다. 그중 일부는 물리, 수학에서 유용하게 쓰이는 공간들이고, 그 공간들이 벡터 공간의 공리를 만족시킨다면, 벡터 공간에서 나온 여러 정리들을 적용시킬 수 있음을 의미한다.

Theorem 21 (Cancellation Law) **x,y,z** $\in V$textbf{x+z=y+z}` 라면, **x=y** 이다.

Corollary 22 (Cancellation Law) **0** 는 유일하다.

각 원소 **x**textbf{-x}` 또한, 각각에 대해 유일하다.

4.2 부분 공간(Sub Space)

대수학에서 대수 구조에 **부분** (Sub)이라는 붙은 용어를 쉽게 찾아 볼 수 있다. 기본적으로 우리가 본 대수 구조들에서도 부분 집합(Sub set), 부분 군(Sub group), 부분 체(sub field) 등등 다양한 부분 대수 구조들이 존재한다. 부분이라는 말은 우리가 말하고 있는 대수 구조가 보다 큰 규모의 대수 구조에 들어가 있다는 것을 말한다. 이 말은 다음 두 가지를 의미한다.

1. 대수 구조를 이루는 집합의 모든 원소가 포함되는 대수 구조의 집합의 원소이다.
2. 부분이라고 부르는 대수 구조는 그 스스로 완전한 대수 구조여야 한다.

예를 들어서 집합을 보자. 집합은 가장 기초적인 대수 구조이다. 연산자 없이 원소들이나 범주만으로 정의된다. 즉, 원소 자체로 그 스스로 완전한 집합을 이룬다. 어떤 집합 B 의 모든 원소가 A 의 원소라면, 이것으로 충분하다. 우리는 이때, 집합 B 를 A 의 부분 집합이라 부르고 이를 다음과 같이 표기한다.

$$B \subset A$$

반면, 연산자가 정의된 대수 구조는 양상이 조금 다른데, 1. 조건을 만족한다 하더라도, 2.를 만족 시키기 위해서는 연산자들이 정의되어야 한다. 이때, 부분 대수 구조가 가지는 연산자는 이 대수 구조를 포함하는, **주 대수 구조가 가지는 연산자** 여야 한다. 즉, 다시 말해 스스로 완전한 대수 구조라는 말의 의미는, 주 대수 구조에서 정의된 연산자에 대해 부분 대수 구조를 이루는 집합의 원소들이 완전한 대수 구조를 이룸을 의미한다. 완전한 대수 구조는 간단하게 판단 할 수 있다. 어느 대수구조의 부분 집합의 원소들은 당연히, 해당하는 대수 구조의 공리들을 모두 만족한다. 즉, 파악해야 하는 부분은 연산자가 **부분 집합에 대해 닫혀있는 지** 와 대수 구조 공리계에 따라 **필수적인 원소의 유무** 만 보면 된다.

우리가 다루고 있는 벡터 공간이라는 대수 구조에서도 이러한 부분 구조를 볼 수 있다. 이를 **부분 공간(sub space)** 라 부른다.

Definition 23 (부분 공간 Sub space) 체 $\mathbb{F}$ 위에 정의된 벡터 공간 $(V, +, \cdot)$ 에 대해 집합 W 가 다음을 만족한다.

- $W \subset V$
- $(W, +_W, \cdot_W)$ 이 벡터 공간을 이룬다.

이때, $(W, +_W, \cdot_W)$ 을 **벡터 공간 ** :math:`(V,+,cdot)` **의 부분 공간** 이라 부른다.

Theorem 24 벡터 공간 $(V, +, \cdot)$ 에 대해 다음이 성립한다.

집합 V 이다.

Theorem 25 벡터 공간 $(V, +, \cdot)$ 의 부분 공간 W_1, W_2 에 대해 다음이 성립한다.

- $W_1 \cap W_2$
- $W_1 + W_2 = \{w_1 + w_2 | w_1 \in W_1, w_2 \in W_2\}$

는 벡터 공간 $(V, +, \cdot)$ 의 부분 공간을 이룬다.

4.3 선형 결합(Linear combination)

Definition 26 (선형 결합 Linear Combination) 체 $\mathbb{F}$, 위에 정의된 벡터공간 Vlambda_1, dots, lambda_n in mathbb{F}` 이고 :math:` bf{v}_1 ,dots , bf{v}_n in V`` 일 때,

$$\sum_{i=1}^{n} \lambda_i \mathbf{v_i}$$

를 $\mathbf{v_i}$ **의 선형 결합** 이라 부른다. 이때, λ_i 를 **선형 결합의 계수** 라 부른다.

3 차원 실수 공간($\mathbb{R}^3$)의 벡터 $\mathbf{a} = (\mathbf{4}, \mathbf{3}, \mathbf{5})$ 를 생각해보자, 일반적으로 각각의 성분들이 x, y, z 축 방향 성분의 크기를 나타내는 것을 잘 알고 있다. 이 축을 나타내는 벡터를 $(\mathbf{e}_i)_{i=1}^3$ 이라 해보자

$$\hat{\mathbf{e}}_1 = (1, 0, 0)$$

$$\hat{\mathbf{e}}_2 = (0, 1, 0)$$

$$\hat{\mathbf{e}}_3 = (0, 0, 1)$$

로 볼 수 있고, 이제 우리는 벡터 $\mathbf{a}$ 를 $\hat{\mathbf{e_i}}$ 의 선형 결합으로 나타낼 수 있다.

$$\vec{\mathbf{a}} = \sum_{i=1}^{3} \lambda_i \mathbf{e_i} = 4 \cdot \hat{\mathbf{e_1}} + 3 \cdot \hat{\mathbf{e_2}} + 5 \cdot \hat{\mathbf{e_3}}$$

이렇게 벡터를 다른 벡터들의 선형 결합으로 표현하는 것은 문제를 풀거나 이해할 때 매우 중요한 도구가 되어준다.

Definition 27 (생성 span) 체 $\mathbb{F}$ 일 때,

$$\text{span}(S) = \{\sum_{i=1}^{n} \lambda_i \mathbf{x_i} : \lambda_i \in \mathbb{F}, \mathbf{x_i} \in S\}$$

를 벡터 집합 S **의 생성** (span)이라 한다.

이러한 생성은 여러 성질을 가진다. 다음은 생성의 성질들이다.

- $\text{span}(\emptyset) = \{\mathbf{0}\}$
- $S \subset \text{span}(S)$
- $A \subset B \rightarrow \text{span}(A) \subset \text{span}(B)$
- $\text{span}(\text{span}(A)) \subset \text{span}(A)$

만약, 벡터공간 V 의 부분 집합 A 에 대해, $A = \text{span}(A)$ 이라면 A 는 벡터공간 V 의 부분 공간이다. 이 역 또한, 성립한다.

Definition 28 (선형 생성 Linear span) 벡터 공간 V 에 대해, 그 부분 집합 S 가 $\text{span}(S) = V$ 를 만족한다면, 벡터 집합 S 가 벡터 공간 V 를 **생성** (spans) 아니면 **선형 생성** (linear generates)한다고 부른다.

4.4 선형 독립(Linear independence)

어떠한 벡터 공간 Vbf{v}` 의 경우 그 벡터 공간 안의 다른 벡터들 $\{\mathbf{v_i}\}_{\mathbf{i=1}}^{\mathbf{n}}$ 과 계수 $(\lambda_i)_{i=1}^n$ 에 대해 다음과 같이 선형 결합으로 표현할 수 있다.

$$\mathbf{v} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \lambda_\mathbf{i} \mathbf{v_i}$$

이때, 계수들 $(\lambda_i)_{i=1}^n$ 가 유일할 필요는 없다. 이는 유일할 수도 유일하지 않을 수도 있다. 이러한 유일성의 유무를 이용해 주어진 벡터 집합의 특성을 정의할 수 있다.

Definition 29 (선형 독립 Linearly Independence) 벡터 공간 V 와 그 부분 집합 S 에 대해, $S = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ 는 다음을 만족할 때, **선형 독립** (linearly independent)이라 한다.

$\forall \mathbf{v} \in V$ 에 대해, 오직 하나의 $(\lambda_i)_{i=1}^n$ 만이 존재해 다음을 만족한다.

$$\mathbf{v} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \lambda_\mathbf{i} \mathbf{v_i}$$

다른 경우, $(\lambda_i)_{i=1}^n$ 가 유일하지 않은 경우에, 이를 S 가 **선형 종속** (linear dependent)이라 한다.

이 정의 외에 다른 정의로도 선형 독립을 정의할 수 있다.

$\mathbf{0} = \sum_{i=1}^N \lambda_i \mathbf{v}_i$ 을 만족하는 $\{\lambda_i\}_{i=1}^N$ 이 유일하게 존재한다.

이 정의는 위의 정의와 동치 관계이다. 이 정의는 실제 어느 집합이 선형 독립인지 판단할 때, 유용하게 쓰인다. 하나의 벡터(영 벡터)의 선형 결합이 유일한지 결정하는 것만으로 독립성을 판별할 수 있기 때문이다.

Theorem 30

- $\emptyset$ 은 선형 독립이다.
- 선형 종속 집합 S 는 $|S| > 0$ 을 만족한다.
- 선형 독립 집합 S 과 벡터, $\mathbf{v} \in V$ 에 대해, $\mathbf{v} \in \text{span}(S)$ 이면, $S \cup \{\mathbf{v}\}$ 는 선형 종속이다.
- 선형 독립 집합 S 과 벡터, $\mathbf{v} \in V$ 에 대해,:math:mathbf{v}notin text{span}(S) 이면, $S \cup \{\mathbf{v}\}$ 는 선형 독립이다.
- $S_1 \subseteq S_2 \subseteq V$ 에 대해, S_1 이 선형 종속이면, S_2 도 선형 종속이다.
- $S_1 \subseteq S_2 \subseteq V$ 에 대해, S_2 이 선형 독립이면, S_1 도 선형 독립이다.

4.5 기저(Bases)

어떤 체 $\mathbf{v}$ 위에 정의된 벡터 공간 V 에서, 내부의 부분 집합 S 의 원소들의 선형 결합으로 벡터 공간 V 의 어느 원소 $\mathbf{v}$ 를 $\mathbb{F}$ 의 원소에 속하는 계수 $(\lambda_i)_{i=1}^n$ 와 함께 표현할 수 있음을 알았다. 또, 이러한 S 를 적절히 잡으면 각각의 벡터의 선형 표현을 유일한 $(\lambda_i)_{i=1}^n$ 로 표현할 수 있음을 알 수 있다. 이를 선형 독립적인 벡터 집합이라 한다. 만약, S 의 원소들이 벡터 공간 전체의 원소를 선형 결합으로 유일하게 나타낼 수 있다면, 이러한 S 를 통해서 벡터 공간을 분류하고 성질을 나타내어 볼 수 있다. 이러한 S 를 **기저** (Basis)라 부른다.

Definition 31 (기저 Basis) 벡터 공간 V 와 그 부분 집합 S 에 대해, S 가 V 의 선형 독립 부분 집합이고,

$\text{span}(S) = V$ 를 만족하면, 이 때 S 를 V 의 **기저** (Basis)라고 부른다.

$$\iff \forall \mathbf{v} \in V, \exists! (\lambda_1, \ldots, \lambda_n) \text{ s.t } \mathbf{v} = \sum \lambda_i \mathbf{v}_i$$

$$S =$$

$$\mathbf{v}_i$$

$$\overset{N}{\underset{i=1}{}}$$

이러한 기저 조건을 만족하는 벡터 집합은 유일하지 않다. 주어진 기저로 부터 다른 기저를 만들수도 있고 상황에 맞게 적절한 기저를 선택할 수도 있다. 다만 각각의 공간에 대해 공통적으로 많이 쓰이는 표준 기저들이 존재한다.

- $\mathbb{F}^n$ 의 표준 기저:
  $\mathbf{e_i}_{i=1}^{n}, \mathbf{e_i} = (0, \ldots, 0, 1, 0, \ldots, 0) = (\delta_{1i}, \ldots, \delta_{(i-1)i}, \delta_{ii}, \delta_{(i+1)i}, \ldots, \delta_{Ni})$
- $M_{m\times n}(\mathbb{F})$ 의 표준 기저:
  $E^{ij}, (1 \le i \le m, 1 \le j \le n)$, $(E^{ij})_{kl} = \delta_{ik}\delta_{jl}$
- $\mathcal{P}_n(\mathbb{F})$ 의 표준 기저: $S =$
  $1, x, x^2, \ldots, x^n$
- $\mathcal{P}(\mathbb{F})$ 의 표준 기저: $S =$
  $1, x, x^2, \ldots,$

그런데 벡터 공간을 이루는 원소가 동일하다 하더라도 체가 달라지면, 벡터 공간의 구조도 달라지게 된다. 예를 들어서 $\mathbb{R}$ 과 $\mathbb{C}$ 를 보면, 이 두 집합은 일반적인 실수 체 위에서 벡터 공간을 형성한다. 이때, 체를 유리수, 복소수로 바꾸면 기저의 수가 달라지는 것을 볼 수 있다.

벡터 공간	표준 기저	기저의 수
$\mathbb{R}(\mathbb{R})$	(1)	1
$\mathbb{R}(\mathbb{Q})$	S	∞
$\mathbb{C}(\mathbb{R})$	$(1, i)$	2
$\mathbb{C}(\mathbb{C})$	(1) or (i)	1

$\mathbb{R}(\mathbb{Q})$ 의 기저가 무한 개인 이유는 실수에 포함되어 있지만, 유리수에는 포함되어 있지 않은 **무리수** (Irrational number) 때문이다. 이 무리수는 유리수들로 나타낼 수 없으며, 무한 개이기 때문에, 이들 하나하나를 모두 기저로 잡아 주어야 한다. 대표적인 무리수로 $\sqrt{2}, \sqrt{3}$ 이 있고, 그 안의 무리수의 일종으로 **초월수** (transcendental number)가 있다. 대표적인 초월수로 π 와 e 가 있다.

Theorem 32 벡터 공간 V 와 그 유한 부분 집합 $S =$
$\mathbf{x_1}, \ldots, \mathbf{x_n}$ 에 대해, 만약 $\text{span}(S) = V$ 라면, 벡터 공간 V 의 기저에 대해, 다음을 만족하는 기저:math:S' 가 존재한다.

Theorem 33 (대체 정리 Replacement Theor) 체 $\mathbb{F}$ 위에서 정의된 벡터 공간 V 와 그 부분 집합 $G \subseteq V$ 에 대해, G 가 V 의 생성 집합이고 $|G| = n$ 이라 하자. V 의 선형 독립 부분 집합 $L \subseteq V$ 가 존재해, $|L| = m$ 일 때, $m \le n$ 이고, G 의 부분 집합 H 가 존재해 다음을 만족한다.

$$|H| = n - m \text{ s.t } \text{ span}(L \cup H) = V$$

Theorem 34 (대체 정리 Corollary 1)

체 $\mathbb{F}$ 위에서 정의된 벡터 공간 V 가 $|G| = n < \infty$ 를 만족하면, V 의 다른 기저 $G' \neq G$ 가 존재할 때,

$$|G'| = |G| = n$$

을 만족한다.

대체 정리는 다시 말해

1. 모든 선형 독립적인 부분 집합은 생성 집합의 원소보다 작거나 같다.
2. 어떠한 선형 독립 부분 집합:math:L 과 생성 부분 집합:math:G 에 대해, G 의 일부 원소를 L 과 합해 새로운 생성 집합을 만들 수 있다.

를 의미한다.

Definition 35 (공간의 차원 Dimension) 벡터 공간 V 의 차원은 V 의 기저 S 의 크기 $|S|$ 를 이 벡터 공간 V 의 **차원** (dimension)이라 한다.

벡터 공간의 차원은 무한할 수도 유한할 수도 있다. 만일 벡터 공간이 유한할 경우 기저가 유한해 유한한 차원을 가진다. 무한할 경우에는 기저가 무한할 수도, 유한할 수도 있다.

차원의 개념이 있다면 대체 공간의 따름 정리 2를 볼 수 있다.

Theorem 36 (대체 정리 Corollary 2) 체 $\mathbb{F}$ 위에서 정의된 벡터 공간 V 가 $\dim(V) = n < \infty$ 일 때, 그 부분 집합 $S \subseteq V$ 에 대해 다음이 성립한다.

1. $\mathrm{span}(S) = V$ 일 때, $|S| \geq n$ 이다. 만일 $|S| = n$ 이면, S 는 V 의 기저이다.
2. S 가 선형 독립 부분 집합일 때, $|S| = n$ 이면, S 는 V 의 기저이다.
3. 모든 선형 독립 부분 집합 S 는 V 의 기저로 확장할 수 있다.
4. 모든 생성 부분 집합 S 은 V 의 기저로 줄일 수 있다.

Theorem 37 벡터 공간 V 와 그 부분 집합 S 에 대해, S 가 유한 부분 집합이고, $\mathrm{span}(S) = V$ 이면, 벡터 공간 V 의 유한 기저 S' 가 존재해, $\dim(V) < \infty$ 가 성립한다.

한가지 들 수 있는 의문점은 우리가 차원을 기저를 통해 정의했다는 점이다. 이 말은 벡터 공간이 기본적으로 기저를 가지고 있다는 가정을 전제로 하고 있다. 그런데 아직 우리는 벡터 공간에 대해, 기저의 존재성을 논하지 않았다. 이는 **초론의 보조 정리(Zoron's lemma)** 라는 선택 공리계와 동치인 정리를 이용해서 증명해야 한다. 이는 현재 단계에서는 공리로써 받아들이고 넘어가도록 하자, 이후 우리가 무한 차원의 공간을 다룰 때, 다시 보도록 하자.

Definition 38 (부분 공간의 차원 Dimension of subspace) 체 $\mathbb{F}$ 위에서 정의된 벡터 공간 V 에 대해, 부분 공간 $W \subseteq V$ 이 있다면, 다음이 성립한다.

$$\dim(W) \leq \dim(V)\&$$

4.6 라그랑주 보간법(응용)

선형 대수학의 중요 개념은 어떤 대상이든지 간에 그 대상의 집합과 연산이 벡터 공간의 공리계를 만족하면 그 대상을 벡터로 보고 여러 정리와 정의들을 응용할 수 있다는 것이다. 이러한 예시로 **라그랑주 보간법(Lagrange interpolation)** 을 볼 수 있다. 보간법(interpolation)이란, 우리가 알고 있는 값 지점들을 이용해 그 지점들 사이의 값을 추정하는 것이다.

라그랑주 보간법은 n 개의 지점을 n 차 다항식을 통해 근사한다. 이때, 쓰이는 다항식을 라그랑주 다항식이라 하는데 다음과 같이 정의된다.

Definition 39 (라그랑주 다항식 Lagrange polynomial) 체 $\mathbb{F}$ 에서 정의된 크기 $n+1$ 의 스칼라 집합 $c_0, c_1, \ldots, c_n$ 에 대해, 크기 $n+1$ 의 다항식 집합 $f_0, f_1, \ldots, f_n$ 과 그 원소를 다음과 같이 정의한다.

$$f_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - c_k}{c_i - c_k}$$

이를 **라그랑주 다항식** (Lagrange polynomial)이라 부른다.

정의에 따라 다음을 알 수 있다.

- $f_i \in \mathcal{P}_n(\mathbb{F})$
- $f_i(c_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

이때, 라그랑주 보간법은 이러한 라그랑주 다항식이 $\mathcal{P}_n(\mathbb{F})$ 의 기저를 만족해 $c_0, c_1, \ldots, c_n$ 점을 모두 지나는 유일한 n 차 다항식을 찾을 수 있다는 것을 기반으로 한다. 이는 벡터 공간의 대체 정리를 통해 증명할 수 있다.

먼저, $f_0, f_1, \ldots, f_n$ 의 선형 독립성을 증명하면,

$$\sum_{i=0}^{n} a_i f_i(x) = 0, \forall x \in \mathbb{F}$$

에 대해, $x \in c_0, c_1, \ldots, c_n$ 이라면, 다음을 알 수 있다.

$$\sum_{i=0}^{n} a_i f_i(c_j) = a_j$$

따라서 선형 독립 조건을 만족하는 계수 $\{a_i\}_{i=0}^{n}$ 은 $a_i = 0, \forall i$ 임을 알 수 있다. 따라서 $f_0, f_1, \ldots, f_n$ 은 $\mathcal{P}_n(\mathbb{F})$ 의 선형 독립 부분집합이다.

$\mathcal{P}_n(\mathbb{F})$ 의 차원=기저 집합의 크기는 $n+1$ 이고 선형 독립 부분 집합 $|f_{i}{}_{i=0}^{n}| = n+1$ 임을 알 수 있다. 여기서 대체 정리의 따름 정리 2-2를 이용하면, $f_{i}{}_{i=0}^{n}$mathcal{P}_n(mathbb{F})` 의 기저 조건을 만족함을 알 수 있다.

따라서 $\mathcal{P}_n(\mathbb{F})$ 에 속한 모든 n 차 다항 함수는 $f_{i}{}_{i=0}^{n}$ 의 선형 결합으로 표현할 수 있다.

이를 이용하면 다음을 보일 수 있는데, n 차 다항 함수 g 에 대해,

$$g = \sum_{i=0}^{n} \lambda_i f_i$$

$$g(c_j) = \sum_{i=0}^{n} \lambda_i f_i = \lambda_j$$

$$g = \sum_{i=0}^{n} g(c_j)_i f_i$$

형태로 유일한 선형 결합 표현을 찾을 수 있고, 이를 이용해, $n+1$ 개의 지점을 지나는 n 이하의 차수를 가지는 유일한 다항 함수를 찾을 수 있다.

이를 **라그랑주 보간법(Lagrange interpolation)** 이라고 한다.

4.7 행렬

여러 추상적인 수학적 대상을 실제 문자로 나타내는 방법은 여러가지가 있다. 함수의 경우 문자 + (변수)를 이용하고 수열은 문자_index 등의 표기를 사용한다. 여러가지 예시가 있지만, 공통적으로 이러한 표기법들은 그러한 대상들의 성질을 잘 표현하고 연산에서 편리성을 보장해준다.

행렬(Matrix) 은 수학에서 자주 쓰이는 표현으로 텐서(tensor), 선형 계(linear system)의 표현, 미분 방정식의 풀이 등등 학부 이상의 고급 수학에서 활발하게 쓰인다.

행렬은 본래 연립 1차 방정식을 편리하게 표기하기 위해 고안되었다. 다음과 같이 4개의 변수를 가지는 방정식 3개를

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_1$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_1$$

간단하게, 다음과 같이 표현 가능하다.

$$bf{A} cdot x = b$$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

이런 방정식을 선형계(Linear system)이라 부른다. 이때, **A** 와 같은 대상을 행렬(matrix)라 부른다. 이와 같이, 행렬은 사각형으로 배열된 원소들의 표현을 의미한다. 각각의 원소들의 위치를 표현하기 위해서 **행(row)** 과 **열(colum)** 을 나타내는 순서쌍 (i, j) 을 사용한다. 행은 사각형에서 가로축을 의미하고, 열은 세로축을 의미한다. 순서쌍에서 i 는 행을, j 는 열을 나타낸다. 각 숫자는 사각형 배열의 왼쪽 위에서부터 행은 아래로, 열은 오른쪽으로 점차 증가하며 세게된다.

일반적으로 행렬의 전체 행과 열의 숫자를 m , n 으로 표기하며 $m \times n$ 행렬이라는 말은 행의 갯수가 m 이고 열의 갯수가 n 개인 행렬을 의미한다.

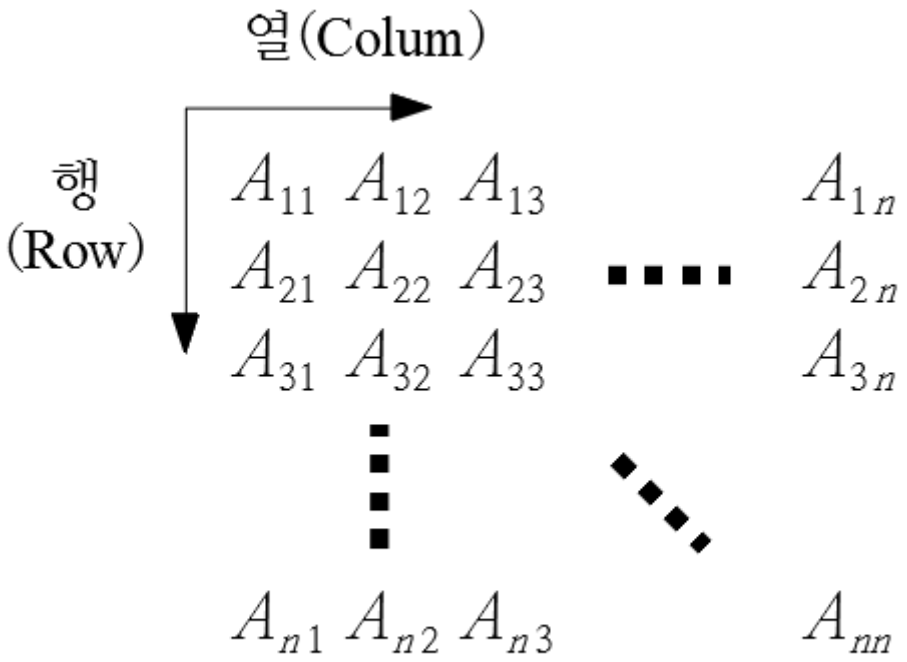
> 행렬 자체는 단순한 사각형 배열이지만, 행렬이 쓰이는 이유인 **행렬의 연산** 을 고려하면, 수학적으로 이를 재정의 할 수 있다. 행렬도 우리가 다룬 대수 구조 처럼 덧셈과 곱셈등의 연산을 정의할 수가 있다. 이러한 행렬의 연산은 행렬을 이루는 원소들의 연산을 통해 만들어진다. 이때, 행렬의 원소들은 적어도 두 덧셈, 곱셈의 연산을 가지고 있어야한다. 따라서, 이를 만족하는 최소한의 대수구조는 환(Ring)이 된다. 이를 이용해 행렬을 다음과 같이 정의할 수 있다.

Definition 40 환 R , 열 $j \in 1, 2, 3, \ldots, n$ 의 순서쌍 (i, j) 에 환의 원소 $A_{ij} \in R$ 를 대응 시키는 함수 $A = A((i, j)) = A_{ij}$ 를 환 R 라 한다.

다시말해, 행렬이란 덧셈과 곱셈이 정의된 원소들을 순서쌍에 대응시킨 함수라 볼 수 있다. 이때, 행렬을 이루는 원소들이 속한 대수 구조에 대해 행렬이 그 위에서 정의되었다라 한다. 일반적으로 행렬을 M 으로 표기하고, $A \in M$ 은 A 가 행렬이라는 뜻이다. 그러나 행렬이 실질적으로 연산을 해야하는 경우에 이는 **같은 대수구조 위에서 정의** 된 행렬에서 의미가 있고, 행렬의 크기(행과 열의 크기)도 행렬의 성질을 결정하는 중요한 요소이므로 이를 표기에 반영해 다음과 같이 표기한다.

$$M_{m \times n}(\mathbb{F})$$

이는 대수구조 $\mathbb{F}$ 위에서 정의된 $m \times n$ 행렬을 의미한다. 따라서 $A \in M_{m \times n}(\mathbb{F})$ 는 A 의 원소들이 대수구조 $\mathbb{F}$ 의 원소들이고, A 의 크기가 $m \times n$ 이라는 뜻이다.



일반적으로 많이 정의되는 공간은 체(Field)이다. 대다수의 문제나 상황에서는 체 위에서 정의된 행렬을 다룬다.

4.7.1 행렬의 연산

기초 연산

체 $\mathbb{F}$ 위에서 정의된 행렬 $\mathbf{A}, \mathbf{B} \in M_{m\times n}(\mathbb{F})$ 과 $c \in \mathbb{F}$ 대해, 행렬의 덧셈, 스칼라 곱 그리고 같음은 다음과 같이 정의된다.

Definition 41 (행렬의 덧셈 Addition of Matrix)

$$(\mathbf{A} + \mathbf{B})_{ij} = (A_{ij} + B_{ij})$$

Definition 42 (행렬의 스칼라 곱 Scalar Multiplication of Matrix)

$$c \cdot \mathbf{A}_{ij} = (c \cdot A_{ij})$$

Definition 43 (행렬의 비교 Equality of Matrix)

$$\mathbf{A} = \mathbf{B}$$
$$\leftrightarrow A_{ij} = B_{ij}, \forall i, j$$

이 연산들과 함께 $M_{m\times n}(\mathbb{F})$ 는 벡터 공간의 공리계를 만족하므로 벡터 공간을 형성한다.

0 행렬은 다음과 같이 정의되고 이는 $M_{m\times n}(\mathbb{F})$ 의 0 벡터가 된다.

Definition 44 (영 행렬 Zero matrix) 행렬 $A \in M_{m\times n}(\mathbb{F})$ 를 영행렬이라고 부른다.

$$A_{ij} = 0_{\mathbb{F}}$$
$$i = 1, 2, \ldots m$$
$$j = 1.2. \ldots n$$

Definition 45 (행렬의 전치 Transpose) 행렬 $\mathbf{A} \in M_{m\times n}(\mathbb{F})$ 에 대해, 행렬 $\mathbf{B} \in M_{n\times m}(\mathbb{F})$ 가 다음을 만족한다.

$$\mathbf{B_{ij}} = \mathbf{A_{ji}}$$

이때, 행렬 $\mathbf{B}$ 를 행렬 $\mathbf{A}$ 의 **전치** (Transpose)라 부르고, $\mathbf{A^t}$ 로 표기한다.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix}$$

$$\rightarrow \mathbf{A^t} = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \\ a_{14} & a_{24} \end{pmatrix}$$

행렬끼리의 곱은 다른 연산에 비해 특이하게 정의된다.

Definition 46 (행렬곱) 행렬 $\mathbf{A} \in M_{m\times n}(\mathbb{F})$, $\mathbf{B} \in M_{n\times p}(\mathbb{F})$ 에 대해, 이 두 행렬의 행렬곱 $\mathbf{C}$ 은 다음과 같다.

$$\mathbf{C} := \mathbf{AB} c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} i = 1, 2, \dots mj = 1, 2, \dots pk = 1, 2, \dots n$$

정의에 의해 $\mathbf{C} \in M_{np}(\mathbb{F})$ 임을 알 수 있다. 이 곱은 우리가 아는 일반적인 곱셈과 다른 점이 많다. 먼저, 정의에 의해 교환 법칙이 성립하지 않는다. 일반적으로 $\mathbf{AB} - \mathbf{BA} \neq 0$ 이고, 특수한 경우 $\mathbf{AB} - \mathbf{BA} = 0$ 를 만족하는 Commuting matrix라 부른다.

벡터와 행렬의 곱은 벡터 $\mathbf{x} \in M_{m\times n}$ 를 m 이나 n 이 1인 행렬임을 알면 행렬곱이라는 것을 알 수 있다. 이 때. $\mathbf{Ax}$ 에서 $\mathbf{A}$ 의 행의 갯수가 $\mathbf{x}$ 의 길이와 같아야한다.

이러한 정의는 행렬을 통해 **선형 변환** (Linear transformation)을 나타내기 위함으로 이러한 행렬 곱을 통해 여러 실제 선형 변환을 실수 연산으로 계산할 수 있다. 실제 선형 변환과의 관련성은 이후의 단원에서 선형 변환을 배우고 알아보도록 하고, 여기서는 이러한 행렬 곱의 성질에 대해 알아보자

- 두 행렬 $\mathbf{A} \in M_{m\times n}(\mathbb{F}), \mathbf{B} \in M_{k\times l}(\mathbb{F})$ 에 대해, 행렬곱 $\mathbf{AB}$ 를 의미한다.
- 두 행렬 $\mathbf{A} \in M_{m\times n}(\mathbb{F}), \mathbf{B} \in M_{k\times l}(\mathbb{F})$ 에 대해, 행렬곱 $\mathbf{C} = \mathbf{AB}$ 는 $M_{m\times l}$ 에 속한다.
- 벡터 $\mathbf{x}$ 와 행렬 $\mathbf{A} \in M_{m\times n}$ 사이의 곱은 열 벡터:math:bf x 에 대해, 이를 $M_{n\times 1}(\mathbb{F})$ 크기의 행렬로 보았을 때, 행렬의 곱 $\mathbf{Ax}$ 과 같다. 행 벡터:math:bf y 의 경우 전치 연산을 통해 가능하다. $\mathbf{Ay^T}$
- 행렬곱은 일반적으로 교환 불가능하다. $\mathbf{AB} - \mathbf{BA} \neq \mathbf{0}$, 만일 두 행렬 $\mathbf{A}, \mathbf{B}$ 가 $\mathbf{AB} - \mathbf{BA} = \mathbf{0}$ 를 만족한다면, $\mathbf{A}, \mathbf{B}$ 가 교환가능하다라 한다. 영어로는 commuting matrix라고 한다.
- $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$
- $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$, $(\mathbf{B} + \mathbf{C})\mathbf{D} = \mathbf{BD} + \mathbf{CD}$
- $c(\mathbf{AB}) = (c\mathbf{A})\mathbf{B}$, $(\mathbf{AB})c = \mathbf{A}(\mathbf{B}c)$
- $(\mathbf{AB})^{\mathbf{T}} = \mathbf{B^T A^T}$

일반적으로 행렬은 실수 체 $\mathbb{R}$ 위에서 정의된 경우를 다루는데, 복소수 체 $\mathbb{C}$bf A in M_{m times n}(mathbb {C})` 에 대해, .. math:

```
\bf {A}^{\dagger}_{ij} = \overline{A}_{ij}
```

를 의미한다. (A 표기도 쓴다.)

4.8 정사각 행렬

행렬 $M_{m\times n}(\mathbb{F})$ 에 대해, $m = n$ 인 경우, 이를 정사각 행렬이라고 부른다.

정사각 행렬에서, 행렬의 대각 성분 중 가장 큰 성분을 주 대각 성분이라 한다.

Definition 47 (주 대각 Diagonal) 정사각 행렬 $\mathbf{A} \in M_n(\mathbb{F})$ 에서

$$A_{ii} \in$$

$$A_{11}, A_{22}, \ldots, A_{nn}$$

i = 1, 2, ⋯n

를 **A** 의 주 대각 성분이라고 하며, diag(**A**) 로 표기한다.

Definition 48 (대각합 Trace) 정사각 행렬 $\mathbf{A} \in M_n(\mathbb{F})$ 에서

$$\sum_{i=1}^{n} A_{ii} \in \text{diag}(\mathbf{A}) i = 1, 2, \ldots n$$

를 **A** 의 **대각합** (Trace)라고 하며, tr(**A**) 로 표기한다.

이 때, 대각합은 다음의 성질을 가진다.

- $\text{tr}(AB) = \text{tr}(BA)$
- $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$
- $\text{tr}(cA) = c \cdot \text{tr}(A)$

이를 이용하면 한가지 재미있는 사실을 보일 수 있다. 우리가 행렬 곱이 교환 불가능함을 보일 때, 다음과 같은 식을 사용했다. $$AB - BA$$ 일반적으로 해당 식은 영행렬이 아니다. 이때, $C = AB - BA$ 인 행렬 C 에 대해, $\text{tr}(C)$ 이 된다. 반대로, $\text{tr}(C) = 0$ 인 행렬은 항상 $AB - BA$ 꼴로 분해할 수 있다. (증명은 나중에 하도록하자)

정사각 행렬의 경우 연산의 항등원이 존재하는 데, 이를 항등 행렬(identity matrix)라고 하며, **I** 로 표기한다. n 차원의 정사각 행렬 공간의 항등 행렬은 $\mathbf{I}_n \in M_n$ 로도 표기한다.

$$\mathbf{I}_n = \begin{pmatrix} 1 & 0 & 0 & & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}$$

Definition 49 (항등 행렬 Identity Matrix) 정사각 행렬 $\mathbf{I} \in M_n(\mathbb{F})$ 에 대해, 다음과 같은 행렬 $\mathbf{I}$ 를 **항등 행렬** (Identity Matrix)이라 하고, $\mathbf{I}_n$ 이라 표기한다.

$$\mathbf{I}_{ij} = \delta_{ij}$$

항등 행렬은 단위 행렬(Unit matrix)라고도 한다. 이 항등 행렬은 행렬곱에 대해 교환 가능하다. 즉,어떤 임의의 행렬 $\mathbf{A} \in M_n(\mathbb{F})$ 에 대해 다음이 성립한다.

$$\mathbf{AI_n} - \mathbf{I_nA} = 0$$

스칼라 행렬은 항등 행렬에 스칼라를 곱한 형태 $c\mathbf{I}_n$ 의 행렬로 행렬 합과 곱에서 스칼라처럼 사용 가능하다.

정사각 행렬끼리의 곱은 곱하기 전과 동일한 차원으로 닫혀있기 때문에, 동일한 행렬의 연속적인 곱을 수행할 수 있다. 이때, 우리가 체에서 차수에대해 한 것처럼 이러한 곱에 대해 비슷한 행렬을 정의할 수 있다. 이를 멱영 행렬(nilpotent matrix)이라 한다.

Definition 50 (멱영 행렬 Nilpotent Matrix) 정사각 행렬 $\mathbf{A} \in M_n(\mathbb{F})$ 에 대해,

$$\exists r \geq 1 \text{ s.t } \mathbf{A}^r = 0$$

인 행렬 $\mathbf{A}$ 을 **멱영 행렬** (Nilpotent Matrix)이라 한다.

앞서 행렬을 정의할 때, 행렬의 전치를 보았다. 어느 $M_{m\times n}$ 행렬의 전치는 $M_{n\times m}$ 행렬로 표현된다. $m = n$ 인 정사각행렬에서 이러한 전치 연산은 같은 공간안으로 되돌아오게 된다. 따라서 행렬의 전치와 행렬을 비교할 수 있는데, 만약 같은 경우 이를 우린 **대칭 행렬(symmetric matrix)** 라 부른다.

Definition 51 (대칭 행렬 Symmetirc Matrix) 정사각 행렬 $\mathbf{A} \in M_n(\mathbb{F})$ 와 $i, j = 1, 2, \ldots n$ 에 대해,

$$A_{ij} = A_{ji}$$

인 행렬 $\mathbf{A}$ 을 **대칭 행렬** (Symmetric Matrix)이라 한다.

제 5 장

선형 변환과 행렬

5.1 선형 변환(Linear Transform)

앞선 단원들에서 우리는 벡터 공간이라는 대수 구조와 그 성질에 대해 배웠다. 주어진 체와 집합 그리고 그 위에서 정의 된 연산은 **벡터 공간** 이라는 특정한 구조를 가진 공간을 이루게 된다. 또한, 이러한 벡터 공간에서 벡터 공간에 속한 특정한 원소들의 **선형 결합** 으로 벡터 공간 전체의 원소들을 나타낼 수 있다는 것을 알 수 있었고, 이를 벡터 공간의 생성 집합, 이러한 생성 집합 중 가장 작은 집합을 **기저** 라고 하였다. 그리고 기저의 갯수를 우리는 벡터 공간의 **차원** 이라 하였다.

어떤 집합과 체, 연산을 가지고 오더라도 벡터 공간 공리계를 만족하는 이상, 이는 벡터 공간이 된다. 공간을 분류하는 것은 어떤 공간이 어떤 공간과 **같은** 구조를 가지는 지, **다른** 구조를 가지는 지 분류함을 의미한다. 이를 위한 표식으로 우리는 모든 공간이 가지는 성질이며, 구조에 따라 다른 값을 가지는 표식을 이용할 것이다. 바로 공간의 **차원** 이다. 모든 벡터 공간은 기저를 가지고 이를 이용해 차원을 구할 수 있다. 그렇다면 차원을 이용해 실질적으로 공간을 어떻게 분류할 수 있을까?

먼저, 분류를 하기에 앞서서 우리가 분류하고자 하는 모든 공간은 동일한 체 위에서 정의되었다는 전제를 깔아두자. 비교하고자 하는 두 공간의 차원이 다르다(=기저의 갯수가 다르다.)면 이것만으로 충분하다. 둘의 구조가 다름을 알 수 있다. 하지만, 두 공간의 차원이 같은 상황에서는 이를 어떻게 분류할 수 있을까? 이를 위해서는

차원이 같은 벡터 공간은 같은 구조를 가지는가

를 답할 수 있어야한다. 이러한 질문에 답을 하기 위해서 이 단원에서 벡터 공간의 **구조를 보존** 하는 특정한 변환인 **선형 변환** 을 소개할 것이다.

결론부터 말하자면 동일 체 위에 정의 된 차원의 크기가 같은 벡터 공간은 같은 구조를 가진다. 이를 이용하면, 특정한 벡터 공간의 연산을 다른 벡터 공간의 연산으로 표현할 수 있다. 이 선언은 아주 중요하다. 왜냐하면, 우린 실제로 원소들을 계산해 실질적인 값으로 만들 수 있는 벡터 공간을 가지고 있기 때문이다. 바로 n -차원 유클리드 공간 $\mathbb{R}^n$ 이다.

이에 더해, 실수 체 위에서 정의된 유한한 차원의 경우, 단순히 원소들을 실수 순서쌍으로 표현할 수 있을 뿐만 아니라, 연산까지도 실수들로 이루어진 특정 표현으로 표현해 낼 수 있다. 바로 **행렬** 이다. 모든 유한 차원의 선형 변환은 행렬로 표현할 수 있고, 모든 행렬은 그 자체로 특정한 선형 변환을 나타낸다.

이 단원에서 우리는 먼저, 행렬의 기초와 구조를 보존하는 선형 변환을 배우고 이러한 선형 변환과 행렬이 동치관계라는 사실을 증명해, 앞으로 우리가 다룰 많은 추상적 대상들을 실수의 연산으로 표현할 수 있음을 보일 것이다.

선형 변환(transform) 과 자주 혼용 되는 표현으로 함수(function) 대응 관계(map) 연산(operator) 등이 있다. 이 책 전반부에서는 변환(transform)이라는 표현을 쓸 것이지만, 양자 역학등의 물리에서는 연산자(operator)라는 표현을 자주 쓸 것이다.

5.1.1 동치 관계

변환을 소개하기에 앞서 **같다** 라는 것이 수학적으로 어떤 의미인지를 정의하고 가도록 하자. 선형 변환을 공부하는 목적은 공간의 구조가 같음을 판별하기 위함인 만큼, 같다라는 의미가 무엇인지 먼저 정의해야 한다. 이는 어느 특정한 **관계** 의 성질을 나타낸다고 볼 수 있다. 수학적으로 같다는 **동치 관계** (equivalence relationship)로 정의할 수 있다.

Definition 52 (동치 관계 Equivalence relationship) 공집합이 아닌, 어느 집합 S 와 그 원소 $x, y, z \in S$ 에 대해 이항 관계 $\sim$ 가 다음을 만족한다.

$$\begin{aligned} x \sim x, \forall x \in S \\ x \sim y \Rightarrow y \sim x, \forall x, y \in S \\ x \sim y, y \sim x \Rightarrow x \sim z, \forall x, y, z \in S \end{aligned}$$

이를 만족하는 $\sim$ 를 S 위에서 정의된 **동치 관계** 라 부른다. 집합을 표기해 $\sim_S$ 로 표기하기도 한다.

동치 관계를 구성하는 3개의 공리는 각각 순서대로, **반사성** (Reflexive), **대칭성** (Symmetric), 그리고 **추이성** (Transitive)으로 부른다. 동치 관계를 가지고 동치 관계인 원소들로 이루어진 집합을 정의 할 수 있다.

Definition 53 (동치류 equivalent class) 공집합이 아닌 집합 S 와 그 위에서 정의된 동치 관계 $\sim$ 에 대해, 어느 원소 $x \in S$ 의 동치류는 다음을 만족하는 S 의 부분 집합이다.

$$[x] := \{y | x \sim y, \forall y \in S\}$$

이러한 동치류는 결국 어느 한 원소와 동치 관계인 원소 집합을 의미하므로 이러한 원소를 표기에 반영해 $[x]$ 과 같이 표기한다. 그런데 동치 관계의 특성상 이러한 대표원은 x 하나만 가능한 것이 아니라, 동치류 안의 모든 원소들이 대표원으로써 사용 될 수 있다. 이 성질은 공체 관계의 추이성으로 바로 유도할 수 있다. 다시 말해, $a \in [x]$ 라면, $[a] = [x]$ 인 것이다. 즉, 정의된 동치 관계로 구분되는 경우에 이들은 사실상 같은 원소들인 것이다.

5.1.2 선형 변환

Definition 54 (선형 변환 Linear transformation) 체 $\mathcal{F}$ 위에 정의된 벡터 공간 V, W 에 대해, 함수 $T : V \to W$ 가 다음을 만족한다.

$$\begin{aligned} \mathbf{x}, \mathbf{y} \in V, c \in \mathcal{F} \\ T(\mathbf{x} + \mathbf{y}) = T(\mathbf{x}) + T(\mathbf{y}) \\ T(c\mathbf{x}) = cT(\mathbf{x}) \end{aligned}$$

이 때, 함수 T 를 $V \to W$ 의 **선형 변환** 이라 한다.

이러한 변환을 통해 보존 되는 벡터 공간의 구조는 일차 종속, 독립, 기저의 수 등이 있다. 이러한 보존 되는 공간의 성질을 우리는 **불변** (Invariant)하다고 한다.

위를 만족하는 선형 변환은 다음을 만족한다.

Theorem 55 (선형 변환의 성질) 선형 변환 $T : V \to W$

- $T(0_V) = 0_W$
- $T(c\mathbf{x} + \mathbf{y}) = cT(\mathbf{x}) + T(\mathbf{y})$
- $T(-\mathbf{y}) = -T(\mathbf{y})$

Corollary 56 (1)

$$T(\sum_{i=1}^{n} a_i \mathbf{x}_i) = \sum_{i=1}^{n} a_i T(\mathbf{x}_i)$$

선형 변환이 공간의 구조를 보존한다고 하였는 데, 기저 또한 보존한다.

즉, 기저의 선형 변환은 그 공간의 또다른 기저를 형성한다. 이러한 기저가 계산하기 쉬운 기저가 아닐 수는 있으나, 별도의 기저를 찾을 필요가 없다는 점에서 매우 편리하다. 이제 이를 보이기 위해서는 몇가지 개념이 추가로 필요하다.

고등학교 때로, 돌아가 함수의 여러 성질들을 생각해 보도록 하자. 함수는 어느 함수를 정의하고자 할 때, 우리는 먼저 함수의 정의역, 공역을 두고, 대응 관계를 정의해 공역 내의 치역을 정의한다.

변환에서도

Kernel = Null space Image

이들은 단순히 공간의 원소가 되는 것이 아니라 각각 벡터 공간 V, W 의 부분 공간을 형성한다.

이들 자체로 벡터 공간의 공리계를 만족하므로 다음과 같이 차원을 정의 할수 있다. Kernel의 차원을 nulity(퇴화 차수), Image의 차원을 rank(계수)라 한다.

함수의 여러성질들을 Kernel과 Image의 용어를 이용해 다음과 같이 새로 정의할 수 있다.

injective surjective bijective

역변환의 존재 유무도 매우 중요한 성질중 하나다.

이러한 대응, 함수의 개념을 좀 더 추상화 시켜서 '사상'(morphism)이라고 한다. 우리가 함수의 성질을 앞에 붙여 "일대일 함수", "일대일 대응 함수"라 하듯이, 이러한 사상도 앞에 성질을 나타내는 접두사를 붙여 부르기도 한다.

T 의 성질	morphism
injective	mono-morphism
surjective	epi-morphism
bijective	iso-morphism
$V = W$	endo-morphism
bijective & $V = W$	auto-morphism

이러한 사상이 존재하는 두 공간에 대해, 두 공간이 서로 **동형** (Isomorphic)이다라 한다.

Definition 57 (벡터 공간의 동형 Isomorphic Vector space) 체 $\mathbb{F}$ 위에 정의된 두 벡터공간 V, W 에 대해, 전단사(bijection) 함수 $T : V \to W$ 가 존재할 때, 함수 T 를 **동형 사상** (Isomorphism)이라 하고 V 가 W 와, W 와 V 가 **동형** (Isomorphic)이라 한다.

동형 관계에 있는 두 공간 V, W 를 다음과 같이 표기한다.

$$V \approx W$$

동형 관계는 동치 관계이기도 하다.

이러한 동형 관계에 있는 두 벡터 공간은 **성질** 이 같다. 이 성질이라는 말은 상당히 포괄적인 언급인데, 공간을 분류하고 그 속에서 벌일 수 있는 여러 연산들이 그대로 동형 관계에 있는 연산으로 보존시킬 수 있다는 뜻이다.

구체적으로 $V \approx W$ 와 그 동형 사상 T , 부분 공간 $S_1, S_2 \subset V$ 에 대해

- $T(S_1 \cap S_2) = T(S_1) \cap T(S_2)$
- $T(S_1 \cup S_2) = T(S_1) \cup T(S_2)$
- $\text{span}(S_1) = V \leftrightarrow \text{span}(T(S_1)) = W$
- $\dim(S_1) = \dim(T(S_1))$

를 만족한다.

동형 공간들이 가지는 의미는 다음과 같다. 우리가 어느 벡터 공간을 연구하고자 할 때, 잘 연구된 벡터 공간이 존재하고, 이와 동형이라면, 연구하고자 하는 대상의 연산들을 우리가 아는 공간의 연산과 bijection들로

표현할 수 있다는 것이다. 이러한 동형을 정의할 때 쓴, **구조를 보존** 하는 변환의 경우 단순히 벡터 공간 위에서 뿐만 아니라 군 그리고 위상 공간에 이르기까지 폭넓게 그리고 중요하게 다루어지는 대상이다. 각각의 공간들의 성질에 따라 여러 가지 변환들이 정의 된다.

그리고 우리에겐 잘 정의되고 연구된 벡터 공간 $\mathcal{R}^n$ 이 존재한다. 만약, 모든 n -차원 벡터 공간이 $\mathbb{R}^n$ 과 동형이라면 우린 $\mathcal{R}^n$ 로 존재하는 모든 n -차원 벡터 공간을 표현할 수 있다.

이는 정말 강력한 선언인데, 정말 유용하게도 참이다.

Theorem 58 (n -차원의 동형 공간) 체 $\mathcal{F}$ 위에서 정의된, $\dim(V) = n \in \mathcal{N}$ 인 벡터 공간 V 에 대해, 다음이 성립한다.

$$\mathcal{F}^n \approx V$$

5.1.3 불변 부분 공간(invariant subspace)

5.1.4 순환 부분 공간(Cyclic subspace)

5.2 행렬

5.3 문제

-

제 6 장

행렬식

행렬식에 대해 배울 때, 공대와 같이 수학적 엄밀함이 필요하지 않은 경우 2차원의 행렬식을 정의하고, 이를 n 차원으로 확장해 재귀적으로 행렬식을 정의한다. 그러나 실제 수학적 엄밀함을 더하면 $k-form$ 을 이용해 정의할 수 있다.

6.1 정의

6.1.1 엄밀한 정의

- $\det(A) = \det(A^T)$
- $\det(AB) = \det(A)\det(B)$

Definition 59 (k-선형 형식 k-linear form)) 체 $\mathbb{F}$ 위에 정의된 벡터 공간 집합 $V_\alpha{}_{\alpha=1}^{k}$ 에 대해, 함수 $f : V_1 \times V_2 \times \cdots \times V_k \to \mathbb{F}$ 가 모든 좌표 각각에 대해 선형일 때, 함수 f 를 **k-선형 형식** (k-linear form)이라 부른다.

만약, $V_\alpha = V, \forall\alpha$ 라면, f 를 V 위에 정의된 k- 선형 형식이라 부르고, $k = 2$ 인 경우

inverse of the matrix with determinant Cramer's rule

elementray Row operation and determinannt Upper triangular matrix determinant = product of diagonal entry Two rows are same (identity) det = 0

A invertible =det(a) != 0

6.1.2 재귀적 정의

2차원 행렬의 행렬식

n 차원 행렬의 행렬식

6.2 성질

- $\det(A) = \det(A^T)$

- $\det(AB) = \det(A)\det(B)$

inverse of the matrix with determinant Cramer's rule

elementray Row operation and determinannt Upper triangular matrix determinant = product of diagonal entry Two rows are same (identity) det = 0

A invertible =det(a) != 0

6.3 응용

제 7 장

대각화(Digonalization)

제 8 장

내적과 노름 공간

제 9 장

기초 위상 수학

제 10 장

함수 공간

함수 공간(Function space)
내적 (Inner product)
내적이 만족해야 할 성질: 선형성, 이항 연산, $mathbb{F}$
적분은 선형성 조건을 만족함
슈바르츠 부등식
직교 확장
베셀 부등식
힐베르트 공간과 바나하 공간

제 11 장

미분 방정식

11.1 선형 미분 방정식

선형 미분 방정식은 방정식 내의 연산이 미분 연산을 포함해 모두 선형적인 미분 방정식을 말한다. 벡터 공간에서 말했다시피, 선형 연산자들은 그 자체로 또다른 벡터공간을 형성하고, 당연히 선행 조건에 따라 덧셈과 스칼라 곱에 대해 닫혀 있다. 즉, 이러한 미분 방정식의 연산은 모두 1개의 선형 연산으로 표현할 수 있다.

어느 한 미분 연산 $_x$ 에 대해,

$$D_x := \frac{\partial}{\partial x}$$
$$D_x^n := \frac{\partial^n}{\partial x^n}$$

$$\alpha_n y^{(n)} + \alpha_{n-1} y^{(n-1)} + \cdots + \alpha_2 y^{(2)} + \alpha_1 y^{(1)} + \alpha_0 y = 0$$

에 대해,

$$(\alpha_n D_x^{(n)} + \alpha_{n-1} D_x^{n-1} + \cdots + \alpha_2 D_x^2 + \alpha_1 D_x + \alpha_0) y = 0$$
$$(\sum_{k=0}^{n} \alpha_k D_x^{(k)}) y = 0$$

$$Ay = 0$$

계수가 굳이 상수일 필요는 없다. 좀 더 일반화 시키면,

$$(\sum_{k=0}^{n} \alpha_k(x) D_x^{(k)}) y = \lambda y$$

11.2 미분 방정식의 해 기저

-> Solution space의 차원

제 12 장

변분법