Глава 1. Идеалы, аффинные многообразия и связь между ними

§1 Аффинные многообразия

1.1 Определение и примеры

**Определение 1.1***: Пусть k – некоторое поле,* *f1 , …, fs – полиномы в* *k[x1, …, xn]. Положим* *.*  *– называется аффинным многообразием, определенным полиномами f1, …, fs.*

1.2 Свойства аффинных многообразий

**Лемма 1** (Свойства аффинных многообразий): *Если* *- аффинные многообразия, то и также являются аффинными многообразиями*.

**Доказательство**: Пусть  и . Мы утверждаем, что

1. Если точка принадлежит , то как функции , так и функции обращаются в нуль в этой точке.
2. Если то все обращаются в нуль в этой точке; значит и все функции обращаются в нуль в . Таким образом, и, аналогично, . Следовательно, . С другой стороны, пусть . Если эта точка принадлежит *V*, то все доказано, если же нет, то для некоторого . Так как функции обращаются в нуль в при всех *j*, то все равны нулю в этой точке. Значит, и

Из этой леммы вытекает, что конечные объединения и пересечения аффинных многообразий являются аффинными многообразиями.

§2 Идеалы

1.1 Определение

**Определение 2.1**: *Подмножество называется идеалом, если выполнены следующие условия:*

1. *;*
2. *Если , то ;*
3. *Если и , то .*

**Определение 2.2**: *Пусть – полиномы в . Положим*

*Множество – идеал.*

**Лемма 2.1**: *Пусть принадлежат кольцу ; тогда множество является идеалом в . Оно называетсяы идеалом, порожденным полиномами , а полиномы – образующими этого идеала или его порождающими элементами.*

**Доказательство**: Прежде всего, , поскольку . Пусть теперь , и . Тогда из равенств

вытекает, что – идеал.

Идеал имеет интерпретацию на языке полиномиальных уравнений. Пусть . Рассмотрим систему уравнений

Из этих уравнений мы можем вывести другие, используя обычные алгебраические преобразования. Так, например, если мы умножим первое уравнение на , второе – на и т.д., а затем сложим произведения, то получим уравнение

,

которое является следствием уравнений первоначальной системы. Отметим, что левая часть этого уравнения принадлежит идеалу , т.е. идеал можно рассматривать как множество всех «полиномиальных следствий» системы .

Идеал *I* называется *конечно порожденным,* если существуют полиномы , такие, что ; при этом множествополиномов называется *базисом* идеала *I.*

**Определение 2.3**: *Пусть – аффинное многообразие. Положим*

*Оказывается, что – идеал.*

**Лемма 2.2**: *Пусть – аффинное многообразие. Тогда – идеал, который мы будем называть идеалом многообразия V.*

**Доказательство.** Ясно, что , так как нулевой полином обращается в нуль на и на *V* в частности. Пусть и . Пусть () – произвольная точка из *V.* Тогда  
Отсюда следует, что *I(V)* - идеал.

Глава 2. Базисы Грёбнера

§1 Введение

1. Основные задачи

1. Задача описания идеала. Является ли произвольный идеал   
    конечно порожденным? Другими словами, верно ли, что для некоторых ?
2. Задача о принадлежности идеалу. Пусть , и пусьб задан идеал . Принадлежит полином *f* идеалу *I* или нет? На геометрическом языке эта задача может быть сформулирована так: содержится ли многообразие в многообразии ?
3. Задача решения полиномиальных уравнений. Описать множество решений в системы полиномиальных уравнений  
   Конечно, это то же самое, что описать аффинное многообразие .
4. Задача неявного представления. Пусть *V –* подмножество в , заданное параметрически:  
   Если – полиномы (или рациональные функции) от переменных , то *V* будет аффинным многообразием или его частью. Задача состоит в том. Чтобы задать *V* полиномиальными уравнениями от переменных .

§2 Упорядочение мономов в

1. Введение

Тщательное рассмотрение алгоритма деления в и алгоритма приведения системы (или матрицы) к ступенчатому виду методом исключения Гаусса показывает, что понятие *упорядочения членов* полинома является ключевым в обоих алгоритмах. Алгоритм деления полиномов от одной переменной имеет дело, как правило, со следующим упорядочением мономов:  
Аналогично, при приведении матрицы к ступенчатому виду мы систематически обращаем в нуль главные элементы, т.е. первые слева ненулевые элементы в строках. На языке линейных систем это означает следующий порядок переменных:  
Каждое уравнение записывается в порядке убывания членов. Более того, в ступенчатом виде уравнения системы записаны в порядке убывания старших (главных) членов.  
  
Отметим, что существует взаимно однозначное соответствие между мономами и *n*-наборами (*n*-векторами) показателей степеней   
 Упорядочение, которое мы определим на , определит и упорядочение на множестве мономов: если на в , то мы будем говорить что .

Так как полином есть сумма мономов, то мы должны уметь расположить его члены в порядке убывания (или по возрастанию). Для этого наше упорядочение должно быть *линейным*. То есть для любой пары мономов должно выполнятся ровно одно из следующих условий

Далее мы должны учесть связь упорядочения с операциями сложения и умножения полиномов. Когда мы складываем полиномы, то после приведения подобных мы просто можем переписать члены суммы в требуемом порядке. Ситуация с произведением более сложная. Дистрибутивность умножения по отношению к сложению позволяет свести задачу к случаю умножения монома на полином.

Мы по требуем, чтобы упорядочение мономов обладало следующим свойством. Если , а – произвольный моном, то . В терминах векторов – показателей степеней это означает, что если в , то для любого , .

Теперь мы можем дать определение.

2. Мономиальное упорядочение

**Определение 2.1**: *Мономиальным упорядочением на*  *называется любое бинарное отношение > на , обладающее следующими свойствами*:

1. > *является линейным упорядочением на .*
2. *Если и* ,  *то .*
3. *> вполне упорядочивает , т.е. любое непустое подмножество в имеет минимальный (наименьший) элемент (по соотношению к упорядочению >).*

**Лемма 2.1(условие вполне упорядоченности)**: *Упорядочение > на вполне упорядочивает это множество тогда и только тогда, когда каждая строго убывающая последовательность элементов из   
обрывается.*

**Доказательство.** Докажем эквивалентное утверждение: > не является вполне упорядочением тогда и только тогда, когда существует бесконечная строго убывающая последовательность элементов из .

Если > не есть вполне упорядочение, то существует непустое подмножество , которое не имеет минимального элемента. Возьмем в качестве произвольный элемент из *S.* Так как он не минимален, то в *S* найдется элемент . Так как не минимален, то в *S* найдется элемент . Продолжая этот процесс, мы получим бесконечную строго убывающую последовательность   
Обратно, если существует такая бесконечная строго убывающая последовательность, то множество является непустым подмножеством в , которое не имеет минимального элемента, т.е. > не является вполне упорядочением.

В качестве примера мономиального упорядочения рассмотрим обычное упорядочение натуральных чисел из :  
Все три условия определения 2.1 выполнены. Следовательно упорядочение мономов из по степени (1) является мономиальным упорядочением.

3. Лексикографическое упорядочение

Нашим первым примером упорядочения n-векторов будет лексикографическое упорядочение (или сокращенно *lex*-упорядочение)

**Определение 3.1**: *(лексикографическое упорядочение). Пусть . Мы говорим, что , если самая левая ненулевая координата вектора положительна. Мы будем писать , если .*

Вот несколько примеров:

1. , так как
2. , так как
3. Обычный порядок переменных является *lex-*упорядочкением. Так как

то .

Работая с полиномами от двух или трёх переменных, мы обозначаем переменные через а не  *.* В дальнейшем мы также будем предпологать, что алфавитный порядок .

Надо проверить, что лексикографическое упорядочение удовлетворяет трем условиям определения 2.1.

**Предложение 3.1**: *Лексикографическое упорядочение на является мономиальным упорядочением.*

**Доказательство**. (i) Тот факт, что – линейное упорядочение, прямо следует из определения и из того, что обычное упорядочение на линейно.  
(ii) Пусть . Тогда самая левая ненулевая координата вектора . Пусть это, например, . Но и   
. Тогда , и самой левой ненулевой координатой опять является .  
(iii) Предположим, что не является вполне упорядочением. Тогда по лемме 2.1 должна существовать строго убывающая бесконечная последовательность

элементов из . Докажем, что это невозможно.

Рассмотрим первые координаты векторов . По определению лексикографического упорядочения они образуют невозрастающаю последовательность неотрицательных целых чисел. Так как вполне упорядочено, то эта последовательность «стабилизируется», т.е. существует такое *k*, что первые координаты векторов одинаковы при .   
Начиная с , будем рассматривать вторые (а затем третьи и т.д.) координаты. Последовательность вторых координат векторов не возрастает; значит, она «стабилизируется». Продолжая это рассуждение, мы можем найти такое *l*, что у векторов равны все координаты. Значит, это одинаковые векторы, что противоречит строгому убыванию последовательности.

В случае лексикографического упорядочения переменная больше *любого монома*, который содержит только меньшие переменные, вне зависимости от его степени. Так, при упорядочении мы имеем . В ряде случаев нам будет необходимо учитывать также степени мономов и сравнивать сначала именно степени. Это можно сделать с помощью градуированного лексикографического упорядочения (сокращенно grlex-*упорядочения*).

**Определение 3.2**: *(градуированное лексикографическое упорядочение). Пусть . Тогда мы говорим, что , если*

Таким образом, grlex сначала упорядочивает по степеням, а если степени равны, то использует лексикографическое упорядочение.

Вот пример:

, так как .

**Определение 3.3**: *(градуированное обратное лексикографическое упорядочение grevlex). Пусть . Тогда мы говорим, что , если  
или и самая правая ненулевая координата вектора отрицательна.*

Пример:, так как .

Чтобы объяснить связь между grlex и grevlex, отметим сначала, что оба эти упорядочения одинаково оценивают степень монома. В случае равенства степеней grlex использует lex-упорядочение, т.е. обращает внимание на самую левую (большую) переменную и «предпочитает» *большую* степень. Напротив, grevlex в случае равенства степеней обращает внимание на самую правую (меньшую) переменную и «предпочитает» меньшую степень.

Посмотрим, как мономиальные упорядочения могут помочь при работе с полиномами. Пусть , и пусть выбрано мономиальное упорядочение >. Тогда мы можем однозначно упорядочить члены полинома *f* в соответствии с >.

Пусть, например, Тогда:

1. При lex-упорядочении мы записываем полином *f* в порядке убывания членов так:
2. При grlex-упорядочении запись *f* такова:
3. При grevlex-упорядочении запись *f* такова:

**Определение 3.4**: *Пусть - ненулевой полином в , и пусть > - мономиальное упорядочение.*

1. *Мультистепень полинома f определяется так:  
   (максимум берется по отношению к >).*
2. *Старший коэффицент полинома f – это*
3. *Старший моном полинома f – это  
   (с коэффицентом 1).*
4. *Старший член полинома f – это*

Пусть, например, , как и выше, и пусть > обозначает lex-упорядочение. Тогда

**Лемма 3.1**:(свойства мультистепени). *Пусть – ненулевые полиномы. Тогда:*

1. *Если , то   
   Если, кроме того, , то указанное неравенство становится равенством.*

Глава 3. Алгоритм деления в

§1. Алгоритм деления полинома от одной переменной

1. Введение и описание алгоритма

Работая с алгоритмами, мы будем использовать «псевдокод», что облегчит нам понимание формальных структур. Описание псевдокода дано в приложении (?). Псевдокод похож на язык программирования Паскаль, и алгоритмы, написанные на нем, легко компьютезируются.

Важнейшей частью алгоритма является понятие «старшего члена» полинома от одной переменной. Вот точное определение.

**Определение 1.1**: *Пусть – ненулевой полином,  
где и (т.е. ). Тогда называется старшим членом полинома f и обозначается .*

Теперь мы можем дать описание алгоритма деления.

**Предложение 1.1**: (алгоритм деления). *Пусть – ненулевой полином. Тогда любой полином может быть записан в виде   
где и либо , либо Более того q и r определены однозначно и имеется алгоритм для их вычисления.*

**Доказательство.** Вот алгоритм вычисления *q* и *r,* записанный в псевдокоде:

Операции, подчиненные оператору циклу WHILE … DO, выполняются, пока выполняется условие, записанное между WHILE и DO; … и … - это операторы определения или переопределения значений и . И и являются *переменными* в этом алгоритме – на каждом шаге их значения меняются.

Приступим к доказательству корректности алгоритма. Заметим, сначала, что равенство выполняется при начальных значениях и . Далее, на каждом шаге после переопределения и это равенство должно выполнятся, потому что

Отметим, что выполнение циклического оператора WHILE … DO прекращается, когда утверждение «» становится ложным, т.е. когда или или . Если алгоритм прекращает работу, он выдает требуемые и .

Осталось доказать, что алгоритм обязательно остановится, т.е. что утверждение между WHILE … DO в какой-то момент станет ложным. Самым важным тут является тот факт, что полином или равен нулю, или имеет степень, меньшую, чем степень полинома *r*. Докажем это. Пусть  
и пусть . Тогда

и степень полинома *r* обязана уменьшиться (или *r* обращается в нуль). Так как степень конечна, то алгоритм останавливается после конечного числа шагов.

2. Основные следствия алгоритма деления в

**Следствие 2.1**: *Пусть*  *– ненулевой полином. Тогда он имеет в k не более чем корней.*

**Доказательство.** Применим индукцию