Глава 1. Идеалы, аффинные многообразия и связь между ними

§1. Аффинные многообразия

1. Определение и cвойства аффинных многообразий

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

**Лемма 1.1** (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. Определение

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

1. *Если , то ;*
2. *Если и , то .*

**Лемма 2.1**: *Пусть принадлежат кольцу ; тогда множество является идеалом в . Оно называется 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 2.2**: *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. Основные задачи об идеалах

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

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.

1. Мономиальное упорядочение

**Определение 2.3**: *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 2.3 (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.

2. Лексикографическое упорядочение

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

**Определение 2.4**: *(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 2.1**: *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 2.5**: *(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 2.6**: *(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 2.7**: *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 2.4** (свойства мультистепени). *Пусть – ненулевые полиномы. Тогда:*

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

§3. Алгоритм деления

1. Алгоритм деления полинома от одной переменной

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

Теперь мы можем дать описание алгоритма деления.

**Предложение 2.2** (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 по . Если , то *f –* ненулевая константа, и утверждение справедливо. Пусть утверждение выполняется для всех полиномов степени , и пусть *f* имеет степень . Если *f* не имеет корней в *k,* то утверждение доказано. Пусть теперь – корень полинома . Поделим *f* на . Тогда по предположению имеем  
, где , так как имеет степень один. Положив в этом равенстве , получим , т.е. , и, значит степень полинома *q* равна .

Мы утверждаем, что любой корень полинома *f*, отличный от *a,* является корнем полинома *q*. Если – корень полинома *f*, то , откуда (так как *k -* поле). По предположению индукции *q* имеет не более корней; значит, *f* имеет не более *m* корней в *k.*

**Следствие 2.2**: *Пусть k - поле. Тогда каждый идеал в может быть представлен в виде для некоторого полинома . Более того, f определен однозначно с точностью до умножения на ненулевую константу из k.*

**Доказательство.** Пусть – некоторый идел. Если , то и утверждение доказано. Пусть теперь , и пусть – ненулевой полином минимальной степени (в множестве полиномов, содержащихся в *I*). Мы утверждаем, что . Включение очевидно, так как *I* - идеал. Рассмотрим теперь полином . В соответствии с алгоритмом деления,   
, где или , или . Так как *I* – идеал, то и, значит . Если , то , что противоречит выбору полинома *f*. Значит , т.е. , что доказывает равенство .

Теперь докажем единственность. Пусть . Так как , то для некоторого полинома *h*. Имеем  
т.е. . Аналогично получаем, поменяв местами , что , т.е. . Из равенства следует, что . Значит, - ненулевая константа.

Идеал, порожденный одним элементом называют *главным идеалом.* Таким образом, ввиду следствия 2.2 мы говорим, что является *областью главных идеалов* или сокращенно ОГИ.

3. Наибольший общий делитель полиномов

**Определение 2.9**: *Наибольщим общим делителем полиномов называется полином h, такой, что*

1. *h делит и , и ;*
2. *если p – некоторый полином, который делит и , и , то р делит h.*

Наибольший общий делитель будет обозначаться через НОД*.* Основные свойства наибольших общих делителей сформулированы в следующем предложении.

**Предложение 2.3**: *Пусть Тогда*

1. НОД *существует и единственен с точностью до умножения на ненулевую константу из k;*
2. НОД *является образующим идеала ;*
3. *существует алгоритм для вычисления* НОД*.*

**Алгоритм Евклида** позволяет вычислить наибольший общий делитель двух полиномов в .

Для начала, дадим необходимые определения. Пусть , . Запишем в виде , где *q* и *r* определены, как в предложении 1.1. Тогда *r* называется остатком от деления . Теперь мы можем дать описание алгоритма Евклида:

Переменными алгоритма являются *h* и *s.* Значением *h* является первый полином в каждом НОД, а значением *s –* второй. Переход от очередного НОД к следующему происходит так же, как и соответствующий переход в цикле алгоритма. Таким образом, на каждом шаге алгоритма   
. Работа алгоритма должна прекратиться, так как степени полинома *s* уменьшаются и в некоторый момент *s*  станет равным нулю. В момент остановки алгоритма , т.е. .

**Определение 2.10**: *Наибольшим общим делителем полиномов называется полином h, такой, что*

1. *h делит ;*
2. *если p – некоторый полином, который делит , то р делит h.*

Такой полином *h* обозначается через *.* Основные свойства наибольшего общего делителя сформулированы в следующем предложении.

**Предложение 2.4**: *Пусть , . Тогда*

1. *существует и определен однозначно с точностью до умножения на ненулевую константу из k;*
2. *порождает идеал ;*
3. *Если , то ;*
4. *существует алгоритм для вычисления .*

**Доказательство.** Доказательство пп. (*i*) и (*ii*) аналогично доказательству тех же пунктов предложению 2.4. Докажем (*iii*). Пусть . Тогда  
Из (*ii*) следует, что

Чтобы доказать существование алгоритма, вычисляющего , нужно объединить п.(*iii*) и алгоритм Евклида.

4. Алгоритм деления в

Выше мы рассматривали алгоритм деления для полиномов от одной переменной. Он может быть применен для решения задачи о принадлежности идеалу. Если , то, для того, чтобы узнать, принадлежит идеалу  
 или нет, мы делим :  
где и или . Мы доказали, что в том и только том случае, когда . Таким образом этот алгоритмический метод пригоден для проверки принадлежности полинома идеалу.

Для решения этой же задачи в случае нескольких переменных необходимо обобщить алгоритм деления алгоритм деления в на общий случай полиномиального кольца . Наша цель – научиться делить полином на полиномы . Это означает научиться представлять в виде  
где «частные» и остаток *r* принадлежат . Чтобы корректно определить остаток, тут будут использованы мономиальные упорядочения.

**Теорема 2.1**: (Алгоритм деления в ). *Зафиксируем некоторое мономиальное упорядочение > на , и пусть – упорядоченный s-набор полиномов из Тогда любой полином может быть записан в виде  
где и или , или r есть линейная комбинация мономов (с коэффицентами из k, ни один из которых не делится ни на один из старших членов . Мы называем r* ***остатком*** *от деления полинома f на F. Более того, если , то*

**Доказательство**. Ниже приведено формальное описание алгоритма:

В этом алгоритме переменная *p* на каждом шаге выполняет роль промежуточного делимого, переменная *r* выполняет роль остатка, а переменные выполняют роль частных. Логическая переменная «естьделение» говорит нам, делится ли старший член промежуточного переменного на какой-либо из . Каждый раз, когда мы находимся в главном цикле может произойти ровно одно из двух событий:

* (Шаг деления) Если никоторый член делит , то алгоритм продолжает работу, как в случае одной переменной.
* (Шаг вычисления остатка) Если никакой из не делит , то алгоритм прибавляет к остатку.

Чтобы проверить корректность алгоритма, мы сначала докажем, что равенство  
выполняется на каждом шаге. Очевидно, что оно выполнено для начальных значений . Пусть на некотором шаге оно имеет место. Если следующим является шаг деления, то некоторый делит и равенство  
показывает, что сумма не изменилась. Так как все остальные переменные остались теми же, то изначальное равенство выполняется и на этом шаге тоже. Если же следующим шагом является шаг вычисления остатка, то меняются и *p*, и *r*, но их сумма остается неизменной, так как   
И опять равенство выполняется на следующем шаге.

Далее, обратим внимание, что алгоритм прекращает работу, когда . В этом случае   
Так как к *r* добавлялись только такие члены, которые не делятся ни на один из , то это означает, что удовлетворяют условиям теоремы 4.1 в случае остановки работы алгоритма.

Осталось доказать, что алгоритм в какой-то момент остановится. Для этого нужно заметить, что каждый раз, когда мы заново вычисляем переменную *p*, или ее мультистепень уменьшается (относительно заданного упорядочения), или *p* обращается в нуль. Чтобы доказать это, предположим сначала, что *p* изменилось в ходе шага деления:

Согласно свойству мультистепени мы имеем  
так что *р* и имеют одинаковые старшие члены. Следовательно, их разность имеет строго меньшую мультистепень (если ). Пусть теперь *р* меняется в ходе шага вычисления остатка:  
Очевидно, что здесь , т.е. в обоих случаях мультистепень уменьшается. Если алгоритм не останавливается, то мы получаем бесконечную строго убывающую последовательность мультистепеней. Но так как > является вполне упорядочением, то это противоречит лемме о свойстве мультистепени. Таким образом, в какой-то момент *p* обратится в нуль, и алгоритм остановится после конечного числа шагов.

Осталось установить связь между Каждый член полинома равен для некоторого значения переменной *р.* Начальное значение *p* есть *f*, и мы только что доказали, что мультистепень *p* строго убывает; значит, . Таким образом , если . Доказательство теоремы закончено.