
pyginv Руководство пользователя

Release 1.1

Блинков Ю.А. и Герд В.П.

8 марта 2006 г.

BlinkovUA@info.sgu.ru и gerdt@jinr.ru

Содержание

1	Введение	3
1.1	Проект GINV	3
1.2	Краткое содержание	3
2	Быстрый старт	4
2.1	Установка	4
	Windows	4
	Unix	4
2.2	Запуск	4
3	Виды входных данных	6
3.1	Полиномы	6
3.2	Модули	6
3.3	Дифференциальные уравнения	6
3.4	Разностные уравнения	7
4	Мономы	8
4.1	Интерфейс	8
4.2	Моном	9
5	Коэффициенты	12
5.1	Интерфейс	12
5.2	Коэффициент	13
6	Полиномы	15
6.1	Интерфейс	15
6.2	Полином	15
7	Критерии	18
7.1	Интерфейс	18
7.2	Обертка	18
8	Инволютивные деления	21
8.1	Интерфейс	21
8.2	Инволютивное деление	21
9	Алгоритмы	22
9.1	Basis	22

А	яЯШКЙХ	25
В	оПХ.ЛЕПШ	26
В.1	оНКХМН.ЛШ	26
В.2	лНДСКХ	27
В.3	дХТТЕПЕМЖХЮКЭМШЕ СПЮБМЕМХЪ	27
В.4	пЮГМНЯРМШЕ СПЮБМЕМХЪ	28
С	Сравнение с другими программами	29

©Copyright 2005, The Ginv Development Team.

1 Введение

1.1 Проект GINV

Открытая система GINV реализующая метод базисов Гребнера для систем уравнений.

Представляет собой реализованный на C++ модуль языка Python для построения базисов Грёбнера идеалов и модулей в полиномиальных, дифференциальных и разностных кольцах.

При построение базисов Грёбнера используется инволютивный подход.

GINV представляет собой "open source".

Исходные тексты программ, установочный пакет языка Python, документация на русском и английском языке доступна по адресу <http://invo.jinr.ru>

1.2 Краткое содержание

2 Быстрый старт

2.1 Установка

Для установки необходимо иметь установленный в операционной системе Python с версией (≥ 2.4). Скачать Python можно с сайта <http://www.python.org>.

Для запуска ginv после установки в нестандартный каталог (например не имея прав администратора) необходимо определить переменную окружения PYTHONPATH. PYTHONPATH представляет собой список каталогов (формат такой же, как у строки PATH), разделенных символом ':' ("Unix") или ';' ("Windows"), в которых будет производиться поиск модулей перед поиском по умолчанию.

Windows

Для установки "Windows" используется уже скомпилированный модуль ginv. Это накладывает некоторые ограничения на тип и разрядность используемого процессора. Модуль ginv собран с кодом i586. В результате возможно значительное снижение производительности, например при использовании 64-битового процессора.

Сборка по "Windows" аналогична сборке под "Unix".

Unix

Для инсталляции под "Unix" требуется библиотека gmp <http://www.swox.com/gmp> собранная с поддержкой C++. Для оптимизации кода под архитектуру компьютера можно внести корректировку опций компилятора C++ в файле 'setup.py'.

Сборка модуля ginv происходит по стандартной схеме для Python

```
python setup.py build
```

Инсталляция в стандартный каталог (и сборка если модуль не был собран)

```
python setup.py install
```

Для установки в нестандартный каталог

```
python setup.py install --prefix=/home/user/pyginv
```

2.2 Запуск

Для проверки правильности установки достаточно запустив Python импортировать модуль командой

```
>>> import ginv
```

Следующий пример позволяет привести систему полиномиальных уравнений к "треугольному виду" и вывести ее на экран

```

import ginv

st = ginv.SystemType("Polynomial")
im = ginv.MonomInterface("Lex", st, ['x', 'y', 'z'])
ic = ginv.CoeffInterface("GmpZ", st)
ip = ginv.PolyInterface("PolyList", st, im, ic)
iw = ginv.WrapInterface("CritPartially", ip)
iD = ginv.DivisionInterface("Janet", iw)

basis = ginv.basisBuild("TQ", iD, \
    ['x^3 - y^2 + z - 1', \
     'y^3 - z^2 + x - 1', \
     'z^3 - x^2 + y - 1'])

for p in basis.iterIB():
    print p

```

Результатом работы программы будет вывод на экран трех полиномов:

```

1395367452523974847088496*x + (-36974012043720606602747)*z^26 + (-18042387149981405949931)*z^25 +
33030642427988789483293*z^24 + 290848876458135913225112*z^23 + 129076392042710083597936*z^22 +
(-219952910543490007651408)*z^21 + (-735043497452966299206159)*z^20 + (-310272794617140912662271)*z^19 +
160633475001590756076807*z^18 + 982342916167428059233558*z^17 + (-20920432695844226462772)*z^16 +
1237879512449968788217761*z^15 + (-649364739492104109294111)*z^14 + 1770783798251395989600640*z^13 +
(-3269570785655865115267646)*z^12 + 1017137392573548569160425*z^11 + (-5380290668716432422480115)*z^10 +
2511318414925313271358736*z^9 + (-5820765926743222198514292)*z^8 + 3916234105006809425362144*z^7 +
(-1271740986960481649374276)*z^6 + 4526571866128344763445768*z^5 + 4225399407124480064647579*z^4 +
(-1702488139911346623054830)*z^3 + 2599321588208462787828165*z^2 + (-6499050198343010356877518)*z +
1138531641035453825071656
1395367452523974847088496*y + (-24263245905633422666089)*z^26 + (-2564101797577402706105)*z^25 +
(-7612716934398042952513)*z^24 + 183708473843508669676888*z^23 + 27382590493064706561728*z^22 +
93382883659052502917872*z^21 + (-445019735963407736728965)*z^20 + (-124580325693827052967653)*z^19 +
(-496913514381305731286115)*z^18 + 686028568407960758274770*z^17 + (-293264534395468068345180)*z^16 +
1754916317739869655913083*z^15 + (-1494161044420719326758149)*z^14 + 3166864848665922491507168*z^13 +
(-4324148866718107366719898)*z^12 + 5118193353820594354340467*z^11 + (-9669988041948198759925505)*z^10 +
8234964392530939187961184*z^9 + (-14800290986146424015670204)*z^8 + 14407709710372641542274656*z^7 +
(-17110238558169309305535500)*z^6 + 19289855195567445653291224*z^5 + (-14071675248685297091156215)*z^4 +
17208495316564910541540422*z^3 + (-9780261032315595871168233)*z^2 + 5684543607089979672443254*z +
(-4606430757804595389204888)
z^27 + (-9)*z^24 + 29*z^21 + 6*z^19 + (-53)*z^18 + 22*z^17 + (-63)*z^16 + 96*z^15 + (-149)*z^14 + 242*z^13 +
(-261)*z^12 + 484*z^11 + (-545)*z^10 + 740*z^9 + (-908)*z^8 + 972*z^7 + (-1220)*z^6 + 1047*z^5 +
(-1045)*z^4 + 943*z^3 + (-535)*z^2 + 422*z + (-216)

```

3 Виды входных данных

Виды входных данных определяют способы задания полиномов и правила работы с ними. Различаются 4 вида: полиномы, модули, дифференциальные уравнения и разностные уравнения. Основное различие их в форме представления момнов. Полиномы могут также иметь различные виды коэффициентов: числовые и параметрические. Входные данные могут представляться в любых допустимых выражениях, в том числе и с дробями. Дробь должна иметь в знаменателе тип данных принадлежащих коэффициенту, который после приведения к общему знаменателю отбрасывается. Тип системы задается классом `SystemType`.

```
class SystemType(type, module=None, rightPart=None)
```

Определяет тип полинома

type: строка определяющая тип полинома

Значение type	Тип системы
"Polynomial"	Полиномиальная система 3.1
"DifferentialEquation"	Система линейных дифференциальных уравнений 3.3
"FiniteDifferenceScheme"	Система линейных разностных уравнений 3.4

module: целое число задающая размерность модуля

rightPart: целое число задающая размерность правых частей равенства, на которые будут получены соотношения определяющие совместность системы, имеет смысл только для модуля.

Класс `SystemType` имеет следующие атрибуты:

type

строка определяющая тип системы (read-only)

module

целое число задающая размерность модуля (read-only)

rightPart

целое число задающая размерность правых частей равенства (read-only)

3.1 Полиномы

Например полином от переменных `['x','y','z']` может представлять собой строку следующего вида

```
p = 'df((x^3+y)^2, x, y, 2) + ((1+x)+y*z*y)*(z^3-9)'
```

В данном выражении `df` представляет собой производную по переменным `x` и дважды по `y`.

3.2 Модули

Модули представляют собой представление в виде списка полиномов

```
m = ['(x^3+y)^2', '1+x', 'y*z*y*(z^3-9)', '0', '1']
```

3.3 Дифференциальные уравнения

Дифференциальное уравнение от зависимых переменных $[u', v']$ и независимых переменных $[x', y', z']$ может представлять собой строку следующего вида

$$d = 'df(x*u, x, 2) + 3*df(v, y, y, y)*y - u + y'$$

3.4 Разностные уравнения

Разностное уравнение от зависимых переменных $[u', v']$, независимых переменных $[t', x']$ и параметров $[tau', h']$ может представлять собой строку следующего вида

$$f = 'T(u, t, 2, x)*h + (T(v, x) - T(v, x, 3))*tau'$$

4 Мономы

В ginv реализованны следующие виды упорядочений:

Упорядочения, совместимые с полной степенью: "TopDegRevLex", "DegRevLex"

Упорядочения с векторизацией вычислений: "TopDegRevLexByte", "DegRevLexByte"

Note: Работа с машинным словом наиболее быстрая для процессора. Векторизация позволяет разместить в машинном слове (для компиляторов C это тип unsigned int) несколько показателей степеней. Тем самым в несколько раз ускоряются некоторые операции с мономом и уменьшаются затраты на память. Это имеет смысл когда показатели степеней ограничены и целесообразно только для упорядочений, совместимых с полной степенью.

Для векторизации необходимо знать где в данном компьютере расположен самый значимый бит: в начале или в конце машинного слова. Это задается макросами C WORDS_BIGENDIAN или WORDS_LITTLEENDIAN. Если они не заданы, то векторизация решает только задачу уменьшения используемой памяти.

Упорядочения, несовместимые с полной степенью: "TopLex", "TopElim", "PotLex", "PotDegRevLex", "PosElim", "Lex", "Elim"

4.1 Интерфейс

```
class MonomInterface(order, systemType, independ, depend=None, varSep=None)
```

Задаёт правила работы с мономом

order: строка определяющая тип упорядочения

Значение order	Тип упорядочения
"TopDegRevLex"	упорядочение для модулей сначала по полной степени, а затем по обратной лексикографии и с "термом старше позиции"
"DegRevLex"	упорядочение сначала по полной степени, а затем по обратной лексикографии
"TopDegRevLexByte"	упорядочение для модулей сначала по полной степени, а затем по обратной лексикографии и с "термом старше позиции" и с использованием векторизации
"DegRevLexByte"	упорядочение сначала по полной степени, а затем по обратной лексикографии с использованием векторизации
"TopLex"	упорядочение для модулей по лексикографии и с "термом старше позиции"
"TopElim"	исключающее упорядочение с "термом старше позиции"
"PotLex"	упорядочение для модулей по лексикографии и с "позицией старше терма"
"PotDegRevLex"	упорядочение для модулей сначала по полной степени, а затем по обратной лексикографии и с "позицией старше терма"
"PosElim"	исключающие упорядочение для модулей с "позицией старше терма"
"Lex"	лексикографическое упорядочение
"Elim"	исключающее упорядочение

systemType: определяет тип системы (см. 3)
independ: список независимых переменных
depend: список зависимых переменных, имеет смысл только для упорядочений поддерживающих модули
varSep: номер разделяющей переменной (зависимой или независимой), имеет смысл только для исключающих упорядочений

Класс MonomInterface имеет следующие методы:

order()
возвращает строку определяющую тип упорядочения
dimIndepend()
возвращает количество независимых переменных
independ()
возвращает список независимых переменных
dimDepend()
возвращает количество зависимых переменных
depend()
возвращает список зависимых переменных
varSep()
возвращает номер разделяющей переменной (зависимой или независимой)

4.2 Моном

class Monom(monomInterface, monom)
Задаёт мономом
monomInterface: определяет интерфейс монома (см. 4.1)
monom: строка инициализирующая моном

Пример значения monom	Тип системы
'1'	единичный моном для любого типа системы
'x*y^4*x'	полиномиальная система 3.1
'df(u, x, y, 4, x)'	система линейных дифференциальных уравнений 3.3
'T(u, x, y, 4, x)'	система линейных разностных уравнений 3.4

Класс Monom имеет следующие методы:

degree()
возвращает сумму степеней переменных монома
dependVar()
возвращает номер зависимой переменной
setZero()
делает степени переменных монома равными нулю
prolong(var, deg=1)
умножает моном на переменную с номером var в степени deg
gcd(monom)
возвращает сумму степеней переменных НОД монома и монома monom.
Мономы должны иметь одинаковый интерфейс.

`lcm(monom)`

возвращает НОК монома и монома `monom`.

Мономы должны иметь одинаковый интерфейс.

`divisibility(monom)`

возвращает `True` если моном делится на моном `monom`, иначе `False`.

Мономы должны иметь одинаковый интерфейс.

`divisibilityTrue(monom)`

возвращает `True` если моном делится на моном `monom` и они не равны друг другу, иначе `False`.

Мономы должны иметь одинаковый интерфейс.

Класс `Monom` может быть аргументом следующих функций:

`str(monom)`

возвращает представление в виде строки монома `monom` согласно типу системы 3.

Аналогично работает команда Python `print`

`cmp(monom1, monom2)`

возвращает 1 если `monom1 > monom2`,

возвращает 0 если `monom1 == monom2`,

возвращает -1 если `monom1 < monom2`.

Мономы сравниваются согласно определенному типу упорядочения. Мономы должны иметь одинаковый интерфейс.

`<, >, <=, >=, ==, !=(monom1, monom2)`

Для этих операций мономы сравниваются согласно определенному типу упорядочения. Мономы должны иметь одинаковый интерфейс.

Моном может быть использован в логических выражениях. Единичный моном дает в логических выражениях `False`, а остальные `True`

`*, /(monom1, monom2)`

Эти операции возвращают произведение и частное мономов. Для выполнения деления `monom1` должен делиться на `monom2`.

Мономы должны иметь одинаковый интерфейс.

`*=(monom1, monom2)`

Эти операции присваивают произведение мономов переменной `monom1`.

Мономы должны иметь одинаковый интерфейс.

`len(monom)`

возвращает число независимых переменных в интерфейсе монома `monom`.

Аналогична методу `dimIndepend` класса `MonomInterface`.

`[(monom, i)`

возвращает степень `i` независимой переменной в мономе `monom`.

Моном представляет собой итератор языка Python:

```
import ginv

st = ginv.SystemType("Polynomial")
im = ginv.MonomInterface("DegRevLex", st, ['x', 'y', 'z'])
m = ginv.Monom(im, "x^3*y*x*z^2")

for d in m: print d,
```

В результате будет распечатан список степеней '4 1 2'.

5 Коэффициенты

В ginv реализованы следующие виды коэффициентов:

Коэффициенты многочлена образуют кольцо: "ITwoParametrModularShort",
"IOneParametrModularShort", "IGmpZ"

Коэффициенты многочлена образуют поле: "IModularShort", "IGmpQ"

Note: В случае поля некоторые операции над многочленом можно сделать более эффективными, например в модулярном случае, если это не требует вычисления НОД.

5.1 Интерфейс

class CoeffInterface(type, systemType, parametr=None, modularShort=None)

Задаёт правила работы с коэффициентом

type: строка определяющая тип коэффициента

Значение type	Тип коэффициента
"GmpQ"	рациональные числа \mathbf{Q} на основе библиотеки GMP
"GmpZ"	целые числа \mathbf{Z} на основе библиотеки GMP
"ModularShort"	числа по модулю простого числа p \mathbf{Z}_p при условии $p^2 <$ машинного слова
"OneParametrModularShort"	полиномиальная арифметика с однопараметрическими коэффициентами и числа по модулю простого числа p \mathbf{Z}_p при условии $p^2 <$ машинного слова
"TwoParametrModularShort"	полиномиальная арифметика с двухпараметрическими коэффициентами и числа по модулю простого числа p \mathbf{Z}_p при условии $p^2 <$ машинного слова
"OneParametrGmpZ"	полиномиальная арифметика с однопараметрическими коэффициентами над кольцом целых \mathbf{Z} на основе библиотеки GMP
"TwoParametrGmpZ"	полиномиальная арифметика с двухпараметрическими коэффициентами над кольцом целых \mathbf{Z} на основе библиотеки GMP

systemType: определяет тип системы (см. 3)

parametr: список параметров, имеет смысл только для коэффициентов с параметрами

modularShort: простое число p при условии $p^2 <$ машинного слова, имеет смысл для коэффициентов с модулярной арифметикой

Класс CoeffInterface имеет следующие методы:

type()

возвращает строку определяющую тип коэффициента

isField()

возвращает True если коэффициенты многочлена образуют поле, иначе False

isPseudo()

возвращает True если коэффициенты многочлена образуют кольцо, иначе False

`parametr()`
возвращает список параметров

5.2 Коэффициент

`class Coeff(coeffInterface, coeff='0')`
Задаёт коэффициент

`coeffInterface`: определяет интерфейс коэффициента (см. 5.1)
`coeff`: строка инициализирующая коэффициент

Класс `Coeff` имеет следующие методы:

`isZero()`
возвращает `True` если коэффициент равен нулю, иначе `False`

`isOne()`
возвращает `True` если коэффициент равен единице, иначе `False`

`setZero()`
делает коэффициент равным нулю

`setOne()`
делает коэффициент равным единице

`gcd(coeff)`
возвращает НОД коэффициента и коэффициента `coeff`.
Коэффициенты должны иметь одинаковый интерфейс.

`diff(par, deg=1)`
возвращает производную коэффициента по параметру номером `par` в степени `deg`.

Класс `Coeff` может быть аргументом следующих функций:

`str(coeff)`
возвращает представление в виде строки коэффициента `coeff` согласно типу системы 3.
Аналогично работает команда Python `print`

`+, -, *, /(coeff1, coeff2)`
Эти операции возвращают сумму, разность, произведение и частное коэффициентов. Для выполнения деления `coeff1` должен делиться на `coeff2`.
Коэффициенты должны иметь одинаковый интерфейс.

`-(coeff)`
возвращает отрицательный коэффициент.

`+=, -=, *=, /=(coeff1, coeff2)`
Эти операции присваивают сумму, разность, произведение и частное коэффициентов. Для выполнения деления `coeff1` должен делиться на `coeff2`.
Коэффициенты должны иметь одинаковый интерфейс.

Коэффициент может быть использован в логических выражениях. Нулевой коэффициент даёт в логических выражениях `False`, а остальные `True`.

Небольшая программа работы с коэффициентами:

```

import ginv

st = ginv.SystemType("Polynomial")
ic = ginv.CoeffInterface("GmpZ", st)
print ic.type()

print ginv.Coeff(ic, "12221965").gcd(ginv.Coeff(ic, "196196196"))
print ginv.Coeff(ic, "121965") + ginv.Coeff(ic, "196196196")

```

В результате будет распечатан следующий результат:

```

GmpZ
7
196318161

```

6 Полиномы

В ginv реализованы следующие представления полиномов:

Полином поддерживает двунаправленные итераторы: "PolyArray" (в реализации)

Полином не поддерживает двунаправленные итераторы: "PolyList"

6.1 Интерфейс

class PolyInterface(type, systemType, monomInterface, coeffInterface)

Задаёт правила работы с полиномами

type: строка определяющая представление полинома

Значение type	Представление полинома
"PolyList"	Представляет собой односвязный список членов полинома
"PolyArray"	Представляет собой массив указателей на члены полинома

systemType: определяет тип системы (см. 3)

monomInterface: определяет интерфейс мономов полинома (см. 4.1)

coeffInterface: определяет интерфейс коэффициентов полинома (см. 5.1)

Класс PolyInterface имеет следующие методы:

type()

возвращает строку определяющую представление полинома

6.2 Полином

class Poly(polyInterface, poly='0')

Задаёт полином

polyInterface: определяет интерфейс полинома (см. 6.1)

poly: строка инициализирующая полином (см. 3)

Класс Poly имеет следующие методы:

length()

возвращает количество членов полинома

degree()

возвращает максимальную сумму степеней переменных монома.

Аналогично degree класса Monom

length()

возвращает количество членов полинома

norm()

возвращает норму полинома

lm()

возвращает лидирующий моном полинома

`lc()`
возвращает лидирующий коэффициент полинома

`setZero()`
делает полином равным нулю

`pp()`
делит многочлен на его содержание

`isPp()`
возвращает True если содержание полинома равно 1, иначе False

`prolong(var, deg=1)`
умножает полином на переменную с номером `var` в степени `deg`

`mult(coeff)`
умножает полином на коэффициент
Коэффициенты полинома и `coeff` должны иметь одинаковый интерфейс.

`mult(monom)`
умножает полином на моном
Мономы полинома и `monom` должны иметь одинаковый интерфейс.

`diff(par, deg=1)`
возвращает производную полинома по параметру номером `par` в степени `deg`.

`reduction(poly)`
выполняет приведение полинома по полиному `poly`.
Лидирующий моном `poly` должен делить лидирующий моном полинома. Полиномы должны иметь одинаковый интерфейс.

`spoly(poly)`
строит S-полином полинома и полинома `poly`.
Полиномы должны иметь одинаковый интерфейс.

Класс `Polu` может быть аргументом следующих функций:

`str(ply)`
возвращает представление в виде строки полинома `ply` согласно типу системы 3.
Аналогично работает команда Python `print`

`cmp(poly1, poly2)`
возвращает 1 если `poly1 > poly2`,
возвращает 0 если `poly1 == poly2`,
возвращает -1 если `poly1 < poly2`.
В сравнении участвуют лидирующие мономы полиномов. Полиномы должны иметь одинаковый интерфейс.

`<, >, <=, >=, ==, !=(poly1, poly2)`
Для этих операций лидирующие мономы полиномов сравниваются согласно определенному типу упорядочения. Полиномы должны иметь одинаковый интерфейс.
Полином может быть использован в логических выражениях. Нулевой полином дает в логических выражениях False, а остальные True

`+, /, *(poly1, poly2)`
Эти операции возвращают сумму, разность и произведение полиномов.
Полиномы должны иметь одинаковый интерфейс.

`+=, /=, *=(poly1, poly2)`
Эти операции присваивает сумму, разность и произведение полиномов переменной `poly1`.

Полиномы должны иметь одинаковый интерфейс.

`len(poly)`

возвращает число членов полинома `poly`.

`[(poly, i)]`

возвращает `i`-ый член полинома.

Полином представляет собой итератор языка Python:

```
import ginv

st = ginv.SystemType("Polynomial")
im = ginv.MonomInterface("Lex", st, ['x', 'y', 'z'])
ic = ginv.CoeffInterface("GmpZ", st)
ip = ginv.PolyInterface("PolyList", st, im, ic)

poly1 = ginv.Poly(ip, "(y^3 - x)^3")
print poly1
poly2 = ginv.Poly(ip, "(y^3 - x)^2 + (x^3 - y)^2")
print poly2
poly1 *= poly2

for (m, c) in poly1: print (m, c),
```

В результате будет распечатан следующий результат:

```
(-1)*x^3 + 3*x^2*y^3 + (-3)*x*y^6 + y^9
x^6 + (-2)*x^3*y + x^2 + (-2)*x*y^3 + y^6 + y^2
(x^9, -1) (x^8*y^3, 3) (x^7*y^6, -3) (x^6*y^9, 1) (x^6*y, 2) (x^5*y^4, -6) (x^5, -1)
(x^4*y^7, 6) (x^4*y^3, 5) (x^3*y^10, -2) (x^3*y^6, -10) (x^3*y^2, -1) (x^2*y^9, 10)
(x^2*y^5, 3) (x*y^12, -5) (x*y^8, -3) (y^15, 1) (y^11, 1)
```

7 Критерии

В ginv реализованы следующие критерии равенства нулю S-полиномов:

Без критериев: "Without"

Частичные критерии Бухбергера: "C1", "CritPartially"

Полные критерии Бухбергера: "C1C2C3", "C1C2C3C4"

Note: Для модульных упорядочений работает только цепочный (второй) критерий Бухбергера.

7.1 Интерфейс

class WrapInterface(type, polyInterface)

Задаёт правила работы с критериями

type: строка определяющая вид критерия

Значение type	Вид критерия
"Without"	Без критериев
"C1"	Первый частичный критерий Бухбергера
"CritPartially"	Первый и второй частичные критерии Бухбергера
"C1C2C3"	Первый и второй частичный критерий Бухбергера
"C1C2C3C4"	Первый и второй частичные критерии Бухбергера

polyInterface: определяет интерфейс полинома (см. 6.1)

Класс WrapInterface имеет следующие методы:

type()

возвращает строку определяющую вид критерия

7.2 Обертка

class Wrap(wrapInterface, poly)

Задаёт обертку для организации работы критериев

wrapInterface: определяет интерфейс обертки(см. 6.1)

poly: полином (см. 6.2)

Класс Wrap имеет следующие методы:

lm()

возвращает лидирующий моном полинома

lm()

возвращает лидирующий моном полинома

ancestor()

возвращает лидирующий моном полинома из которого данный полином получен серией продолжений

poly()

возвращает полином

`multi()`
 возвращает количество мультипликативных переменных
`degProlong()`
 возвращает степень переменной по которой полином был получен продолжением из другого полинома
`lm()`
 возвращает лидирующий моном полинома
`isProlong()`
 возвращает True если является ли полином продолжением другого полнома, иначе False
`buildProlong()`
 возвращает список степеней переменных по которым данный полином был продолжен

Небольшая программа работы с обертками полиномов после построения базисов Грёбнера:

```

import ginv

st = ginv.SystemType("Polynomial")
im = ginv.MonomInterface("DegRevLex", st, ['x', 'y', 'z'])
ic = ginv.CoeffInterface("GmpZ", st)
ip = ginv.PolyInterface("PolyList", st, im, ic)
iw = ginv.WrapInterface("CritPartially", ip)
iD = ginv.DivisionInterface("Janet", iw)

def printWrap(w):
    print "      lm =", w.lm()
    print "  ancestor =", w.ancestor()
    print "    poly =", w.poly()
    print "    multi =", w.multi()
    print " degProlong =", w.degProlong()
    print " isProlong =", w.isProlong()
    print "buildProlong =", w.buildProlong()

basis = ginv.basisBuild("TQBlockLow", iD, \
    ['x^3 - y^2 + z - 1', \
    'y^3 - z^2 + x - 1', \
    'z^3 - x^2 + y - 1'])

for i in [0, 6, 12]:
    printWrap(basis[i])
  
```

В результате будет распечатан следующий результат:

```

    lm = x^2*y^2*z^3
    ancestor = z^3
    poly = x^2*y^2*z^3 + (-1)*x^2*y^2 + x*y^2*z + x^2*z^2 +
(-1)*x*y*z^2 + x^2*y + (-1)*x*y^2 + x^2 + (-1)*x*y + (-1)*y^2 + z + (-1)
    multi = 1
    degProlong = 0
    isProlong = True
    buildProlong = [1, 1, 0]
    lm = x^2*y^3
    ancestor = y^3
    poly = x^2*y^3 + (-1)*x^2*z^2 + (-1)*x^2 + y^2 + (-1)*z + 1
    multi = 2
    degProlong = 0
    isProlong = True
    buildProlong = [1, 0, 0]
    lm = z^3
    ancestor = z^3
    poly = z^3 + (-1)*x^2 + y + (-1)
    multi = 1
    degProlong = 0
    isProlong = False
    buildProlong = [1, 1, 0]

```

8 Инволютивные деления

В `ginv` реализованы следующие виды инволютивных делений:

Основаны на дереве Janet: "Janet", "JanetLike"

Note: Дерево Janet позволяет за время $O(d)$, где d сумма степеней монома, находить инволютивного делителя

8.1 Интерфейс

`class DivisionInterface(type, wrapInterface)`

Задаёт правила работы с инволютивным делением

`type`: строка определяющая тип инволютивного деления

Значение <code>type</code>	Тип инволютивного деления
"Janet"	деление Janet
"JanetLike"	деление JanetLike, очень похоже по структуре на Janet, но может гораздо более компактное представление на ненульмерных идеалах

`wrapInterface`: определяет вид применяемых критериев (см. 7.1)

Класс `DivisionInterface` имеет следующие методы:

`type()`

возвращает строку определяющую тип инволютивного деления

8.2 Инволютивное деление

Предполагается реализовать возможность построения инволютивного деления.

9 Алгоритмы

Для построения базиса системы полиномов используется следующая функция:

`basis(algorithm, divisionInterface, system)`
 возвращает объект класса `Basis` представляющий инволютивный базис системы `system`
`algorithm`: строка определяющая используемый алгоритм

Значение <code>algorithm</code>	Используемый алгоритм
"TQ"	алгоритм TQ из статьи ... работает на всех видах упорядочения
"TQDegree"	алгоритм TQDegree из статьи ... работает на упорядочениях, совместимых с полной степенью (см. 4)
"TQBlockHigh"	алгоритм TQBlockHigh из статьи ... работает на упорядочениях, совместимых с полной степенью (см. 4)
"TQBlockLow"	алгоритм TQBlockLow из статьи ... работает на упорядочениях, совместимых с полной степенью (см. 4)

`divisionInterface`: определяет интерфейс инволютивного деления (см. 8.1) используемого при построении базиса

`system`:

список представляющий собой полиномы представленные объектами класса `Poly` или строк их инициализации (см. 6.2)

9.1 Basis

`class Basis`

Представляет собой инволютивный базис. Может быть сконструирован только функцией `basis`.

Класс `Basis` имеет следующие методы:

`lengthIB()`

возвращает количество элементов инволютивного базиса

`lengthGB()`

возвращает количество элементов базиса Грёбнера

`numCriterion()`

возвращает количество примененных критериев равенства нулю S-полиномов при построении базиса

`numSpoly()`

возвращает количество вычисленных S-полиномов при построении базиса

`numReduction()`

возвращает количество выполненных приведений полиномов при построении базиса

`userTime()`

возвращает затраченное пользовательское время при построении базиса

`sysTime()`

возвращает затраченное системное время при построении базиса

`realTime()`

возвращает затраченное астрономическое время при построении базиса

`hilbertPolynomial()`

возвращает строку представляющую полином Гильберта инволютивного базиса

`iterStatistics()`

возвращает итератор для просмотра статистики работы инволютивного алгоритма. Представляет собой кортеж из 4 элементов:

- время шага
- количество элементов в множестве T
- количество элементов в множестве Q
- количество элементов в множестве Q имеющих минимальный лидирующий мономом

`iterIB()`

возвращает итератор для просмотра инволютивного базиса представленного объектами класса `Poly` (см. 6.2)

`iterGB()`

возвращает итератор для просмотра базиса Грёбнера представленного объектами класса `Poly` (см. 6.2)

`iterLmIB()`

возвращает итератор для просмотра лидирующих мономов инволютивного базиса представленного объектами класса `Моном` (см. 4.2)

`iterLmGB()`

возвращает итератор для просмотра лидирующих мономов базиса Грёбнера представленного объектами класса `Моном` (см. 4.2)

`iterLmGB()`

возвращает итератор для просмотра лидирующих мономов базиса Грёбнера представленного объектами класса `Моном` (см. 4.2)

`find(monom)`

ищет инволютивный делитель в базисе. `monom` может представлять собой строку инициализации монома или представлять собой объект класса `Моном` (см. 4.2).

Мономы в базисе и `monom` должны иметь одинаковые интерфейсы.

`nf(poly)`

выполняет приведение `poly` по базису с точностью до лидирующего монома. `poly` может представлять собой строку инициализации монома или представлять собой объект класса `Poly` (см. 6.2).

Полиномы в базисе и `poly` должны иметь одинаковые интерфейсы.

`isNf(poly)`

выполняет проверку приведения `poly` по базису с точностью до лидирующего монома. Возвращает `True`, иначе `False` `poly` может представлять собой строку инициализации монома или представлять собой объект класса `Poly` (см. 6.2).

Полиномы в базисе и `poly` должны иметь одинаковые интерфейсы.

`nfTail(poly)`

выполняет полное приведение `poly` по базису без редукции лидирующего монома. `poly` может представлять собой строку инициализации монома или представлять собой объект класса `Poly` (см. 6.2).

Полиномы в базисе и `poly` должны иметь одинаковые интерфейсы.

`isNf(poly)`

выполняет проверку приведения `poly` по базису без редукции лидирующего монома. Возвращает `True`, иначе `False` `poly` может представлять собой строку инициализации монома или представлять собой объект класса `Poly` (см. 6.2).

Полиномы в базисе и `poly` должны иметь одинаковые интерфейсы.

Класс `Basis` может быть аргументом следующих функций:

`len(basis)`

возвращает число элементов инволютивного базиса `basis`.

`[(basis, i)]`

возвращает i -ый член инволютивного базиса представленный объектами класса `Wrap` (см. 7.2)

Базис представляет собой итератор языка Python:

```
import ginv

st = ginv.SystemType("Polynomial")
im = ginv.MonomInterface("Lex", st, ['x', 'y', 'z'])
ic = ginv.CoeffInterface("GmpZ", st)
ip = ginv.PolyInterface("PolyList", st, im, ic)
iw = ginv.WrapInterface("CritPartially", ip)
iD = ginv.DivisionInterface("Janet", iw)

basis = ginv.basisBuild("TQ", iD, \
    ['x^3 - y^2 + z - 1', \
     'y^3 - z^2 + x - 1', \
     'z^3 - x^2 + y - 1'])

for w in basis: print w.lm(),
```

В результате будет распечатан список лидирующих мономов инволютивного базиса `'x y z^27'`.

А ЯШКЙХ

В оПХЛЕПШ

оПХЛЕПШ ЯНДЕПФЮРЯЪ Б ОЮОЙЕ ‘examples’

В.1 оНКХМНЛШ

дКЪ ГЮОСЯЙЮ ОНКХМНЛХЮКЭМНЦН ОПХЛЕПЮ ДНЯРЮРНВМН НРЙПШРЭ Б idle
‘examples.py’ ХКХ МЮАПЮРЭ Б ОЮОЙЕ ОПХЛЕПНБ ЙНЛЮМДМСЧ ЯРПНЙС

```
python example.py
```

бШАНП ОПХЛЕПЮ ЛНФМН НЯСЫЕЯРБХРЭ ХГ ЯКЕДСЧЫЕЦН ЯОХЯЙЮ:

```

.....
# assur44.xml.gz      ducos7_3.xml.gz    hf855.xml.gz      quadfor2.xml.gz
# aubry2.xml.gz      ducos7_5.xml.gz    hietarinta1.xml.gz  quadgrid.xml.gz
# augot.xml.gz       ducos8.xml.gz      hunecke.xml.gz     rabmo.xml.gz
# benchmark_D1.xml.gz eco10.xml.gz        i1.xml.gz          rbpl24.xml.gz
# benchmark_i1.xml.gz eco11.xml.gz        ilias12.xml.gz     rbpl.xml.gz
# boon.xml.gz        eco12.xml.gz        ilias13.xml.gz     redcyc5.xml.gz
# butcher8.xml.gz     eco7.xml.gz         ilias_k_2.xml.gz    redcyc6.xml.gz
# butcher.xml.gz      eco8.xml.gz         ilias_k_3.xml.gz    redcyc7.xml.gz
# camera1s.xml.gz     eco9.xml.gz         issac97.xml.gz      redcyc8.xml.gz
# caprasse.xml.gz     el44.xml.gz         jcf26.xml.gz        redeco10.xml.gz
# cassou.xml.gz       el50.xml.gz         katsura10.xml.gz    redeco11.xml.gz
# chandra4.xml.gz     extcyc4.xml.gz      katsura6.xml.gz     redeco12.xml.gz
# chandra5.xml.gz     extcyc5.xml.gz      katsura7.xml.gz     redeco7.xml.gz
# chandra6.xml.gz     extcyc6.xml.gz      katsura8.xml.gz     redeco8.xml.gz
# chemequs.xml.gz     extcyc7.xml.gz      katsura9.xml.gz     redeco9.xml.gz
# chemequ.xml.gz      extcyc8.xml.gz      kin1.xml.gz         reif.xml.gz
# chemkin.xml.gz       f633.xml.gz         kinema.xml.gz       reimer4.xml.gz
# cohn2.xml.gz         f744.xml.gz         kotsireas.xml.gz    reimer5.xml.gz
# cohn3.xml.gz         f855.xml.gz         kul0.xml.gz         reimer6.xml.gz
# comb3000s.xml.gz     f966.xml.gz         lanconelli.xml.gz   reimer7.xml.gz
# comb3000.xml.gz      fabrice24.xml.gz    lichtblau.xml.gz    reimer8.xml.gz
# conform1.xml.gz      filter9.xml.gz      liu.xml.gz          rose.xml.gz
# cpdm5.xml.gz         geneig.xml.gz       lorentz.xml.gz      s9_1.xml.gz
# cyclic5.xml.gz        hairer1.xml.gz      matrix.xml.gz       solotarev.xml.gz
# cyclic6.xml.gz        hairer2.xml.gz      mckay.gls50mod.xml.gz sparse5.xml.gz
# cyclic7.xml.gz        hairer3.xml.gz      mckay.xml.gz        speer.xml.gz
# cyclic8.xml.gz        hairer4.xml.gz      mickey.xml.gz        tangents.xml.gz
# d1.xml.gz            hawes4.xml.gz       morgenstern.xml.gz   test.xml.gz
# des18_3.xml.gz       hcyclic5.xml.gz     noon5.xml.gz         uteshev_bikker.xml.gz
# des22_24.xml.gz      hcyclic6.xml.gz     noon6.xml.gz         vermeer.xml.gz
# dessin1.xml.gz       hcyclic7.xml.gz     noon7.xml.gz         vernov1.xml.gz
# dessin2.xml.gz       hcyclic8.xml.gz     noon8.xml.gz         virasoro.xml.gz
# discret3.xml.gz      heart.xml.gz        noon9.xml.gz         wang16.xml.gz
# dl.xml.gz            hemmecke.xml.gz     pinchon1.xml.gz      wright.xml.gz
# ducos10.xml.gz       hf744.xml.gz        puma.xml.gz

folder = os.path.join(".", "polynomial")
fname = "cyclic7.xml.gz"
.....

```

НРПЕДЮЙРХПНБЮБ Б МХФМЕИ ЯРПНЙЕ ХЛЪ ТЮИКЮ. ЕЦН НОХЯЮМХЕ ВШБНДХРЯЪ
МЮ ЩЙПНОМ ЙНЛЮМДНИ

```
print description
```

РЮЙФЕ СГМЮРЭ Н ОПХЛЕПЕ ЛНФМН ОПНЯЛНРПЕБ ЕЦН БН БМСРПЕММЕИ ОЮОЙЕ
'polynomial'. ОПХЛЕПШ УПЮМЪРЯЪ Б xml-ТЮИКЮУ ЯФЮРШУ ЮПУХБЮРНПЛ gzip.

В.2 ЛНДСКХ

В.3 ДХТТЕПЕМЖХЮКЭМШЕ СПЮБМЕМХЪ

В.4 ПОГМНЯРМШЕ СПЮБМЕМХЪ

С Сравнение с другими программами

All times in seconds. Programms have been run on machine 2xOpteron-242 (1.6 Ghz) with 4Gb of RAM, running Gentoo Linux 2004.3 and compiled with gcc-3.4.2. Description of criterions can be found in "paper"section.

means, that example is not computed due to memory overflow.

JB - results of programm by Yanovich D.A.

ginv ? implementation to Python C++ program have been run on machine Turion-3400 (1.8 Ghz) with 2Gb of RAM, running Gentoo Linux 2005.1 and compiled with gcc-3.4.4

Magma V2.12-17 on evariste (dual processor, AMD Opteron, 2200 MHz, 16 GB)

Example	JB	ginv (deg)	ginv (max)	ginv (min)	Singular	Magma (F4)	Magma V2.12-17
assur44	10.35	14.20	6.33	6.4	600.88	4.56	1.13
butcher8	1.06	1.02	0.38	0.39	2.52	4.68	1.22
chemequs	0.67	0.61	0.57	0.6	1.57	12.80	1.84
chemkin	17.83	16.87	10.95	9.95	1.57	32.34	7.64
cohn3	76.72	107.14	30.21	25.47	475.84	37.73	9.68
cpdm5	1.78	1.57	1.69	1.68	2.38	0.69	0.18
cyclic6	0.12	0.19	0.14	0.14	0	0.09	0.02
cyclic7	58.72	60.94	68.59	65.28	*	6.64	1.82
cyclic8	12056.24	14046.26	5826.18	4424.96	*	235.73	57.67
d1	8.77	12.58	1.99	2.08	8.94	28.49	1.70
des18_3	0.19	0.18	0.19	0.19	0	1.81	0.44
des22_24	0.68	0.62	0.77	0.79	0.96	1.37	0.31
discret3	23322.8	20956.31	12642.49	13521.65	*	33658.09	3089.44
dl	270.17	278.89	80.77	89.52	44.79	14.57	2.79
eco8	0.40	0.44	0.44	0.46	0	0.20	0.06
eco9	3.22	5.60	4.99	5.08	4.03	1.25	0.29
eco10	52.56	56.70	65.71	68.06	157.44	7.07	1.75
eco11	765.98	741.74	718.53	679.3	*	62.33	12.43
eco12	4083.00	8824.61	11224.11	11493.49	*		
extcyc5	1.35	1.53	1.46	1.37	2.93	0.37	0.11
extcyc6	324.70	184.49	276.06	155.64	*	45.36	12.02
extcyc7	*	*	*	*	*	8242.00	1819.4
f744	4.88	7.71	2.22	2.68	2.42	1.47	0.31
f855	132.97	139.79	37.64	38.45	546.69	48.63	7.76
fabrice24	108.52	116.77	8.2	7.7	6912.94	9.45	1.68
filter9	20.97	5.76	1.13	1.6	30.06	80.04	13.16
hairer2	62.91	108.17	126.69	125.43	902.54	92.07	17.85
hairer3	1.96	0.92	0.32	1.4	1072.46	*	*
heyclic7	64.17	53.87	65.81	73.0	4574.47	6.26	1.67
heyclic8	6024.97	4316.59	*	7560.99	*	229.70	55.33
hf744	22.17	8.58	7.18	11.26	7.25	1.39	0.30
hf855	2157.88	534.08	806.51	988.38	2057.03	48.15	7.70
hietarinta1	0.77	0.71	0.38	0.53	6.78	2.63	0.46
il	98.24	122.36	58.29	58.21	61.71	55.07	6.89
ilias13	1167.18	5851.97	3013.1	2469.62	56913.46	336.21	70.35
ilias_k_2	323.59	669.68	445.51	270.21	40401.62	55.41	13.46
ilias_k_3	452.32	846.19	1162.7	622.14	26015.41	90.67	21.80
jcf26	224.96	211.24	16.44	14.65	741.20	31.64	5.11
katsura7	2.15	1.77	2.08	1.98	3.10	0.72	0.19
katsura8	27.48	24.66	28.8	27.09	32.18	4.7	1.28
katsura9	337.52	294.59	340.45	311.98	703.45	33.47	8.70
katsura10	4790.55	4983.11	7220.29	6204.95	16547.71	287.38	69.33
kin1	15.18	20.32	7.11	7.11	50.56	45.33	5.65
kotsireas	6.33	37.94	4.93	4.27	9.25	3.45	0.90
noon6	0.97	1.29	1.27	1.29	0	0.60	0.15
noon7	28.87	32.58	37.52	38.52	1.16	4.93	1.24
noon8	1552.26	2292.84	3322.62	3152.57	16.02	43.65	10.37
noon9	84152.10				218.31	422.65	116.40
pinchon1	10.37	0.04	0.01	0.01	8.04	4.09	0.94
rbpl	210.94	177.51	173.8	173.98	1191.37	38.33	8.59
rbpl24	108.78	116.78	8.23	7.7	6862.68	9.62	8.59
redcyc6	0.16	0.17	0.13	0.14	0	0.10	0.02
redcyc7	913.75	1048.69	48.19	48.61	*	5.73	1.65
redco10	18.51	18.66	23.91	22.4	6.96	2.33	0.63
redco11	178.32	187.36	253.34	228.41	65.02	14.56	3.96
redco12	1735.95	2172.75	4666.8	3385.97	591.96	101.51	27.44
reimer5	0.22	0.36	0.34	0.38	1.05	0.74	0.19
reimer6	9.69	21.60	24.19	23.96	*	42.13	10.61
reimer7	719.37	3808.91	4756.4	4314.12	*	5216.53	1196.37
virasoro	9.69	8.90	10.96	10.68	36.25	1.72	0.47