ПРИКЛАДНАЯ МАТЕМАТИКА, МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

***

УДК 517.917

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАБОТЫ С МАТРИЦАМИ ДИРАКА SUSY_DIRACUS

Люлинский М. Х.[a,1]

[a] Казанский федеральный университет, г. Казань, 420008, Россия.

Описан программный комплекс для работы с матрицами Дирака SUSY_Diracus — набор вычислительных инструментов, позволяющий ускорить работу специалиста по суперсимметрии. Описание функций сопровождено примером их использования. Кратко рассмотрены принципы построения и работы этих функций. Приведены примеры нестандартного применения инструментов комплекса.

Ключевые слова: алгебра Клиффорда, матрицы Дирака, Maplesoft, суперсимметрия, суперпространство, супералгебра, супердифференциальный оператор.

SOFT FOR DIRAC'S MATRICES MANIPULATION — SUSY_DIRACUS PACKAGE

Lyulinsky M. Kh.[a,1]

[a] Kazan Federal University, Kazan, 420008, Russia.

Package SUSY_Diracus is discribed. It is equipment usefull for sciencist of supersymmetry. The description of the functions is accompanied by an example of their use. The principles of construction and operation of these functions are briefly considered. Examples of non–standard application of the tools of the complex are given.

Keywords: Clifford Algebra, Dirac Matrices, Maplesot, Supersymmetry, Superspace, Superdifferential operator, Superalgebra.

PACS: 34D08, 93C15
DOI: 10.17238/issn2226-8812.2022.2.54-69

Введение

В современной теоретической физике вычисления с привлечением аппарата клиффордовых алгебр ведутся в широких масштабах. Благодаря трудам таких авторитетных и опытных специалистов в области компьютерной алгебры как В.А. Ростовцев, уже в 70-х годах прошлого столетия для подобных работ стали появляться автоматизированные комплексы [1].

В современных работах часто приходиться иметь дело с выражениями, состоящими из объектов разной математической природы таких как: комплексные числа, элементы различных алгебр, элементы векторных и тензорных пространств над такими алгебрами. Например: в [2] на функции

$$f=\sum_k\sum_{l_1\dots l_k} a_{i,l_1\dots l_k}\prod_{n=0}^{k}\left(\left(\prod_{q=1}^{l_n}\gamma^{l_q}\right)C^{-1}\right)_{\alpha_n\beta_n}\theta^{\alpha_0}\dots\theta^{\alpha_n}\theta^{\beta_0}\dots\theta^{\beta_n} \tag{I}$$

[1] E-mail: miklul@rambler.ru

где $a_{i,l_1\ldots l_k}$ — функции от x^a, $\quad a=0\ldots 3$, $\theta^\alpha\theta^\beta+\theta^\beta\theta^\alpha=0$ (образующие алгебры Грассмана), $(\gamma^a)^\alpha_\beta$ — гамма-матрицы Дирака и

$$C=i\gamma^0\gamma^2$$

действуют операторы вида:

$$M_{ab}=x_a\partial_b-x_b\partial_a+\frac{1}{2}\theta^\alpha(\gamma_{ab})^\beta_\alpha\partial_\beta;$$

что приводит к громоздким выражениям. В вычислении метрики суперпространства де Ситтера используется супердифференциальный оператор [2]

$$K_0=2x_0x^b\partial_b-x^2\partial_0+\left(x_a\left(\gamma_0\gamma^a\right)^\tau_\mu\theta^\mu-\frac{1}{3}\mathbb{C}^\epsilon_{\mu\nu\gamma}\left(\gamma^0\right)^\tau_\epsilon\theta^\mu\theta^\nu\theta^\gamma\right)\partial_\tau+$$
$$+\frac{1}{12}\big(\mathbb{C}^\tau_{\mu\nu\xi}\left(\gamma_0\gamma^bC^{-1}\right)_{\tau\eta}+\left(\gamma_0\gamma_5C^{-1}\right)_{\eta\mu}\left(\gamma_5\gamma^bC^{-1}\right)_{\nu\xi}-\left(\gamma_0\gamma_5C^{-1}\right)_{\eta\xi}\left(\gamma_5\gamma^bC^{-1}\right)_{\nu\mu}+$$
$$+\left(\gamma_0\gamma_5C^{-1}\right)_{\eta\nu}\left(\gamma_5\gamma^bC^{-1}\right)_{\xi\mu}\big)\theta^\eta\theta^\mu\theta^\nu\theta^\xi\partial_b$$

что приводит к выражениям, содержащим несколько тысяч слагаемых. Несмотря на обилие компьютерных средств, выбрать инструмент для подобных задач непросто. Однако системы компьютерной алгебры постоянно совершенствуются. Прекрасно зарекоммендовала себя система *Cadabra* [3]. Однако ей пока далеко до мощи *Wolfram Mathematica* и *Maplesoft Maple*. Необходимость решения описанных выше задач суперматематики привела к созданию комплекса SUSY_Diracus. Наш комплекс построен на базе системы Maple 2016 и успешно работает в более новых версиях. В отличие от проекта [4, 5], здесь есть возможность реализации работы в алгебре Клиффорда с произвольной сигнатурой. В данном случае комплекс настроен на лоренцеву сигнатуру $+---$. Реализован механизм сверток по немым спинорным индексам. Терминология комплекса совпадает с терминологией, принятой в работах [2], [6], что делает его особенно привлекательным для изучения супермногообразий. Если рассмотреть недавно появившийся пакет *GammaMaP* для *Wolfram Mathematica* [7], можно заметить, что в нем отсутствуют инструменты работы с представлением γ-матриц в линейном пространстве. Но в задачах подобных решаемым в [2] в них возникает серьезная необходимость. Здесь эти функции реализованы в полной мере и весьма остроумно (7.10). В отличии от пакета *GammaMaP* мы написали расширение функции $diff$ (8.1) на суперслучай, можно видеть, что комплекс с такими возможностями представлен впервые и хорошо подходит на роль производственной базы для "индустриального метода написания суперсимметричных лагранжианов..." [8], опираясь на фундаментальные симметрии [9] физических систем. Благодаря таким инструментам, к громоздким вычислениям можно приступать с большей уверенностью [10].

Мы надеемся, что наша работа продолжит направление [1], [11]. Комплекс состоит из сорока четырёх процедур выстроенных по принципу многоуровневой обработки данных (рис. 1).

Для работы с нашим комплексом его нужно подключить:

$[>read\quad "SUSY\_Diracus.mpl"$

Необходимые для работы файлы доступны по адресу: www.miklul.ru/Maple. Загрузите всё в один каталог, откройте *Start.mw* и начните работать. Вы можете сохранить вашу работу под любым именем.

1. Постановка задачи

Пусть задано математическое выражение, включающее в себя матрицы Дирака $(\gamma^a)^\beta_\alpha$, например, вида (I). Чтобы упростить такое выражение, требуется произвести соответствующие свертки и коммутации.

Для рассмотрения этой задачи, необходимо описать полный граф состояний вычислительного процесса. Ограничимся вычислением многочлена, слагаемыми которого служат произведения многочленов. В этом случае вычисление начинается с алгебраических действий, т.е. с раскрытия скобок и т.д. В полученном многочлене анализируется каждый моном на предмет возможности

сверток. Сами свертки нужно провести в соответствии с порядком индексов, а затем выполнить необходимые коммутации гамма–матриц. Каждый пункт определяет уровень обработки выражения.

Основной возможностью, позволившей нам быстро реализовать алгоритм работы с матрицами Дирака, стала способность среды Maple создавать алгебраические объекты неалгебраическими методами.

Прежде чем реализовать алгоритм, нужно разобраться в способностях исполнителя. В нашем случае исполнителем будет среда Maple 2016. Сначала мы приведем краткие описания двух встроенных функций, позволивших нам реализовать свой алгоритм.

2. Subs

Subs команда подстановки подвыражения в выражение.

subs(x=a,expr)

subs([S1,...,Sn],expr)

subs[inplace](x=a,expr)

subs[inplace]([S1,...,Sn],expr)

Первая форма команды подстановки *subs* заменяет x на a в выражении *expr*. Следует отметить, что эта команда аналогична команде *Eval*. Простые операции из ряда замен символа на его значение в формуле, как правило, следует сделать с помощью команды *Eval*. Различия между этими двумя командами в том, что при работе с *Eval* старая переменная остается связанной в данном выражении.

Вторая общая форма команды подстановки *subs* делает замены указанные первом аргументе в *expr*. Каждый из $S1, ..., SN$ представляет из себя равенство $x_i = a_i$ или список равенств заключенные в фигурные или квадратные скобки. Замены выполняются последовательно, начиная с $S1$. Замены в рамках списка выполняются одновременно; если есть несколько замен на то же имя, используется первый.

Параметр *Inplace* применяется только при замене в *expr* типа *rtable* (таких как *Array*, *Matrix*, или *Vector*), без функции индексации. Когда *Inplace* указан, вход *rtable* обновляется на месте. В этом случае меняется именно оригинальное выражение *expr*, а не создаётся его копия. Действие замены не включает упрощение выражения. В тех случаях, когда желательно упростить выражение, следует использовать команду *Eval*.

3. Описание функции op

$op(i, e)$

$op(i..j, e)$

$op(e)$

$op(list, e)$

$nops(e)$

Функция *op* раскладывает выражения на составные части. Прелесть этой функции в том и состоит, что в результате мы получаем набор деталей исходного выражения, которыми в дальнейшем можно свободно манипулировать. Начнем с того, что для некоторых структур данных определена опция $op(0, e)$. Эта опция открывает колоссальные возможности для программирования в **Maple**. В таблице ниже приведены результаты применения *op* к выражениям разных типов и конструкций. Так для функций, $op(0, e)$ возвращает имя функции. Для индексных имен, $op(0, e)$ это имя без ндексов.

Конструкция выражения e	$op\,(e)$	$op\,(0,e)$
Адитивность $a+b,\qquad a-b$	a,b	‘+‘
Мультивность $a*b,\qquad \frac{a}{b}$	a,b	‘ . ‘
a^b	a,b	‘ ^ ‘
$a=b$	a,b	$=$
$f\,(a,\ldots,c)$	$a,\ldots,c$	f
$a_{b,\ldots,n}$	$b,\ldots,n$	a
$\frac{df(x)}{dx}$	$f\,(x)\,,x$	$diff$
$\int f\,(x)\,dx$	$f\,(x)\,,x$	int
a	a	$symbol$

Выражение e в контексте функции op представляет собой объект, состоящий из членов (операндов) e, связанных знаками математических операций, которые в соответствиями со своими приоритетами задают уровни вложенности.

В простейшем случае функция op возвращает список частей e, связанных наимение приоритетной операцией. Так если op имеет только один аргумент, она возваратит результаты, представленные во 2-м столбце таблицы. Это равносильно вызову, $op(1..nops(e),e)$. Если первый аргумент в op список, то элементы списка относятся к операндам e с возрастающими уровнями вложенности. Это просто краткая форма записи. Выражение $op([A1,A2,...,],e)$ возвращает тот же результат, что и $op(An,op(...,op(A2,op(A1,e))...))$, но является более быстрым для выполнения. Когда первый параметр представляет собой список, последний элемент списка может быть интервалом. Как говорилось выше, это всего лишь краткая форма записи. Выражение $op([A1,A2,...,an1..an2],e)$ возвращает тот же результат, что и $op(an1..an2,op(...,op(A2,op(A1,e))...))$, но является более эффективным для выполнения.

$nops$ — функция возвращает количество таких деталей или операндов в выражении. Если первый аргумент функции op целое положительное число i, то возвращается i-й операнд e. Если операнд не существует, возникает ошибка. Если первый аргумент в op является отрицательным целым числом i, то в результате возвращается: $op(nops(e)+i+1,e)$.

Для рядов, $op(0,e)$ является x — переменная разложения ряда. Вычисление $op(0,series(exp(x),x=a))$, возвращает x, а не a. Для всех других структур данных, $op(0,e)$ возвращает само e. Если значения операндов селекторов не в диапазоне -$nops\,(e)$ т.е. $[-1...nops(e)]$, возникает ошибка. Если первый аргумент в op является интервалом целых чисел от i-го до j-го $[i..j]$, то результат представляет собой последовательность i-го до j-го членов e. Отрицательные целые числа в диапазоне рассматриваются, как описано выше; к ним добавляется $nops(e)-1$.

4. Способ представления произведений гамма–матриц

Нам нужно записать выражения вида:

$$\sum_i A_i \prod_m \left(\gamma^{a^i_{m_1}}\gamma^{a^i_{m_2}}\ldots\gamma^{a^i_{m_i}}\right)^{\beta_m}_{\mu_k} \prod_k \theta^{\mu_k}$$

Поскольку выражение в скобках имеет индексы, естественно его записать конструкцией *имя-индекс*. Здесь вместо имени должно стоять выражение. В свою очередь выражение должно сохранять порядок множителей. Для решения этого вопроса естественно было бы полагать реализацию в виде:

- Строки,

- Вектор-матрицы,
- Множества,
- Полиарной функции.

Со строками сложно работать, вектор-матрицы оказались неупотребительны системой в данной роли, множества не сохраняют порядок. Четвертый вариант в сочетании с командой op оказался неожиданно элегантным решением. Такой функции было дано имя Ω. В такой записи выражения принимают вид:

$$\sum_i A_i \prod_m \Omega \left(\gamma^{a^i_{m_1}} \gamma^{a^i_{m_2}} \dots \gamma^{a^i_{m_i}}\right)^{\beta_m}_{\mu_k} \prod_k \theta^{\mu_k}$$

5. Принципы построения системы SUSY_Diracus

В основу построения нашей системы положены принципы объектно-ориентированного программирования [12, 13], которые в нашем случае выражаются в специализации функций, что в свою очередь порождает структуру уровней рис. 1 (закрашенные поля). Это значит, что каждая из задач, представленных на уровне интерфейса, решаются не одной отдельной функцией, а группой или конвейером функций, которые вызывают одна другую как представлено на рис. 1. Интерфейсные функции принимают в качестве аргументов выражения полиномиального вида и непосредственно сами представляют собой обычные циклы суммирования результатов обработки мономов, которые доставляются функциями уровня одночленов. На уровне одночленов решаются вопросы привлечения сил клиффордова уровня и т.д.

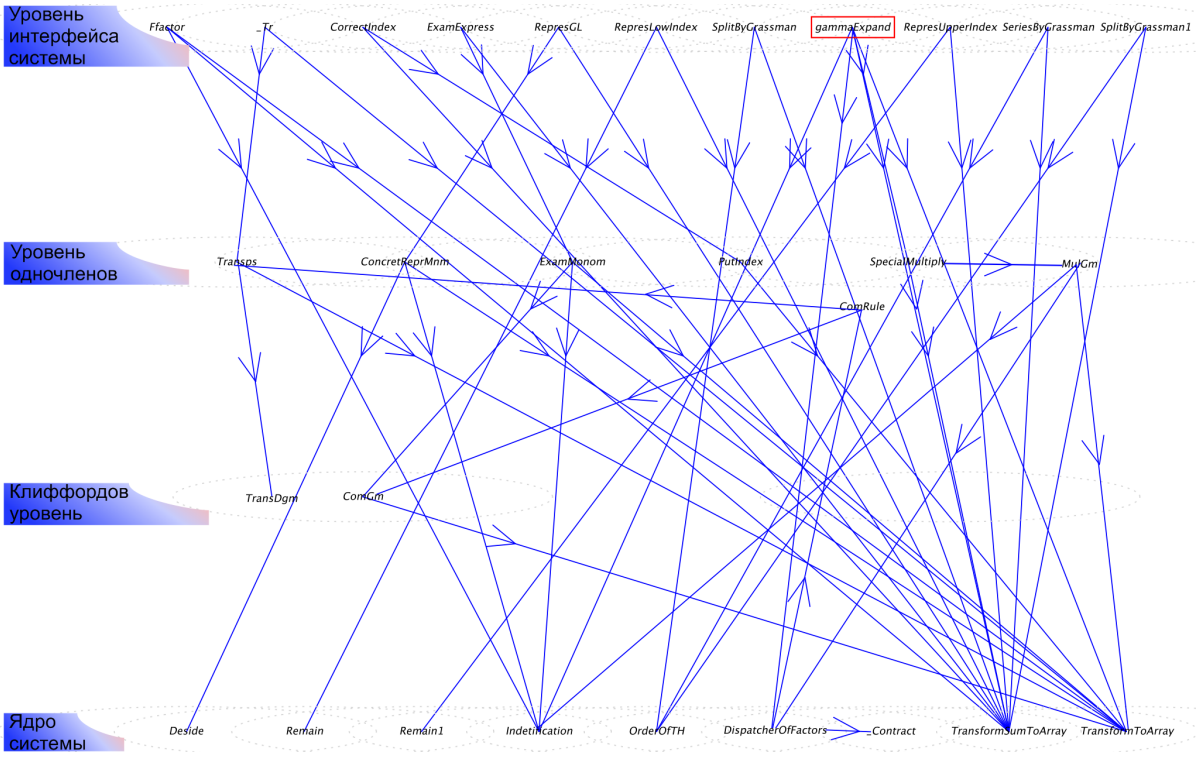


Рис. 1. Граф вызовов.

Согласно сказанному выше сложилась и архитектура программного комплекса. Такая архитектура позволяет легко масштабировать систему и быстро адаптировать её для других алгебр Клиффорда. В нижней строке перечислены функции, представляющие ядро системы. О них будет рассказано в конце. О функциях в других строках будет рассказано в соответствующих разделах. Впрочем, об их назначениях несложно догадаться по их именам.

6. Общие правила и соглашения

При работе с пакетом SUSY_Diracus необходимо принять несколько соглашений.

Есть несколько символов, которые не нужно использовать в качестве переменных. Они обрабатываются функциями пакета специальным образом. Это: $\Omega, \gamma, \delta, C, C1, \eta\eta, \theta$.

- γ — обозначает гамма-матрицу Дирака $\gamma^0, \gamma^1, \gamma^2, \gamma^3, \gamma_5$,
- θ — Грассманова переменная,
- δ — символ Кронеккера,
- C^1 — обозначает $i\gamma^0\gamma^2$,
- $C1 = C^{-1}$ — обозначает $i\gamma^2\gamma^0$,
- Ω — имя формальной функции, оформляющей произведение $\gamma^{a_1}...\gamma^{a_n}$, например, $\Omega(\gamma_1, \gamma_3)$,
- $\eta\eta$ — тензор Минковского.

Эти символы не должны встречаться на позициях индексов и более ни в какой другой роли. Все бозонные и спинорные индексы при координатных переменных считаются верхними. В выражениях Maple:

$[> x_a, \theta_c, \delta_{\xi,\eta}, \Omega(\gamma_b)_{\alpha,\beta}$

индексы a, b, c, β и η считаются верхними, α и ξ нижними. Индексы a и b назовём *бозонными*, а α, β, η, ξ и c -спинорного типа или *спинорными* в соответствии с местом их расположения. Это выражение эквивалентно следующей математической записи [2,14,15]:

$$x^a, \theta^c, \delta^\eta_\xi, (\gamma^b)^\beta_\alpha$$

Обозначения $\Omega(\gamma_5)_{\alpha,\beta}$ — есть $(\gamma_5)^\beta_\alpha$, т.е. в выражениях $\Omega(\dots)_{\dots,\dots}$, $\delta_{\dots,\dots}$ первый индекс считается нижним, второй — верхним. Исключение: $\Omega(C1)_{\alpha,\beta} - \left(C^{-1}\right)_{\alpha\beta}, \Omega(C)_{\alpha,\beta} - C^{\alpha\beta}$. Если в выражении произведения встречаются повторяющиеся индексы спинорного типа, по ним предполагается суммирование.

7. Интерфейс

7.1. gammaExpand

gammaExpand(A)

Процедура *gammaExpand* — главная комманда программного пакета комплекса. Она раскрывает все скобки и выполняет все свёртки спинорных индексов при γ^a. Ответ формулируется в терминах $\delta, \gamma^0, \gamma^1, \gamma^2, \gamma^3$. В каждом мономе гамма-матрицы выстраиваются в порядке возрастания их бозонных индексов. Спинорные индексы расставляются в лексографическом порядке.

Примеры:

$[> gammaExpand(\theta_\mu * \Omega(\gamma_0, \gamma_3)_{\mu,\tau} * \Omega(C)_{\tau,\xi})$;

$$I\Omega(\gamma_2, \gamma_3)_{\mu,\xi}\theta_\mu$$

Полная обработка исходного выражения осуществляется таким алгоритмом:

[1] С помощью антисимметричной матрицы $C = C^{\alpha\beta}$ можно поднимать спинорные индексы α, β, ... , а с помощью обратной к ней матрицы $C^{-1} = {C^{-1}}_{\alpha\beta}$ опускать -, согласно формулам ([2] стр.79):

$$\theta^\alpha = \theta_\beta C^{\beta\alpha}; \theta_\alpha = (C^{-1})_{\beta\alpha}\theta^\beta = -(C^{-1})_{\alpha\beta}\theta^\beta.$$

.

$[> S := ... :$

$[> S := SimplifyByTh(expand(S)) :$

$[> S := CorrectIndex(S, [\xi 1, \xi 2, \xi 3, \xi 4]) :$ ♯ если только этих индексов нет в исходном выражении.

$[> S := gammaExpand(S) :$

$[> S := ExamExpress(S) :$

$[> S := LikeGammaMember(expand(S)) :$

$[> S := gammaExpand(S) :$

Эти команды рекоммендуется оканчивать двоеточием, чтобы не загружать рабочее поле промежуточными вычислениями.

7.2. CorrectIndex

$CorrectIndex\,(A, w)$

Процедура $CorrectIndex$ заменяет спинорные индексы в полиноме A. w — это множество символов, не фигурирующих в качестве связанных переменных в скрипте и (**важно**) невстречающихся[2] в вырыжении A.

7.3. LikegammaMember

$LikegammaMember\,(A)$

В выражении могут встречаться одинаковые по смыслу члены, (отличающиеся лишь немыми индексами). Так как Maple трактует их как набор символов, он не распознаёт их как подобные. Для решения этого вопроса была разработана процедура $LikegammaMember$, которая выявляет подобные Гамма–члены, делая соответствующие замены индексов. **ВНИМАНИЕ**: индексы исходного выражения не должны содержать символа "ω", т.к этот символ используется самой функцией $LikegammaMember\,(A)$. Это условие всегда можно выполнить, используя функцию *subs* (см. выше). Особенностью алгоритма этой функции является обратный ход логики. Все остальные функции обратывают многочлены, обрабатывая каждый одночлен. Сдесь же работы начинается с осмотра всего иходного многочлена.

Идея алгоритма основывается на создании *хеша*[3] или словаря произведений γ-матриц. Это реализуется с помошью системы конструкций $hashmset : -new$ и $array$.

Например: выражение

$[> \Omega(\gamma_0, \gamma_3)_{\mu,\tau}\theta_\mu + \Omega(\gamma_0, \gamma_3)_{\alpha,\tau}\theta_\alpha$

останется без изменений тогда как:

$$[> LikegammaMember\left(\Omega(\gamma_0, \gamma_3)_{\mu,\tau}\theta_\mu + \Omega(\gamma_0, \gamma_3)_{\alpha,\tau}\theta_\alpha\right);$$

$$2\Omega(\gamma_0, \gamma_3)_{\omega 112,\tau}\theta_{\omega 112}$$

. Поэтому лучше:

$$[> CorrectIndex\left(LikegammaMember\left(\Omega(\gamma_0, \gamma_3)_{\mu,\tau}\theta_\mu + \Omega(\gamma_0, \gamma_3)_{\alpha,\tau}\theta_\alpha\right), [\mu, \tau]\right);$$

$$2\Omega(\gamma_0, \gamma_3)_{\mu,\tau}\theta_\mu$$

.

7.4. SimplifyByTh

$SimplifyByTh\,(A)$

Процедура SimplifyByTh убирает из выражения члены содержащие (порядок по θ^α)$> Dim$.

[2]Это является крупной недоработкой создателей комплекса, которые просят за это прощения у всех пользователей и обязуются исправить в скором будущем.

[3]см., например, стр. 123, Р. Шварц, Т.Феникс, Изучаем PERL, 5-е издание, изд. Символ, Ст-Петербург-Москва, 2009г.

7.5. Ffactor

Ffactor (A)

Эта функция выносит общий множитель за скобки. Стандартная функция factor, к сожалению, не справляется с выражениями нашего вида.

Примеры:

$[> Ffactor(-2*I*\kappa*t*\Omega(\gamma_1,\gamma_2,C1)_{\alpha,\mu}*CQ2*CQ1-2*I*\kappa*y*\Omega(\gamma_0,\gamma_1,C1)_{\alpha,\mu}*CQ2*CQ1+2*I*x*\Omega(\gamma_0,\gamma_2,C1)_{\alpha,\mu}*\kappa*CQ2*CQ1-2*\kappa*z*\Omega(\gamma_5,C1)_{\alpha,\mu}*CQ2*CQ1);$

$$2*I*CQ2*CQ1*(I*z*\Omega(\gamma_5,C1)_{\alpha,\mu}-t*\Omega(\gamma_1,\gamma_2,C1)_{\alpha,\mu}+x*\Omega(\gamma_0,\gamma_2,C1)_{\alpha,\mu}-y*\Omega(\gamma_0,\gamma_1,C1)_{\alpha,\mu})*\kappa$$

7.6. Вертикальное перемещение спинорных индексов. RepresLowIndex и RepresUpperIndex

RepresLowIndex (A,w)

RepresUpperIndex (A,w)

Возмем произвольную гамма–матрицу $(\gamma^a)^{\beta}_{\alpha}$. Мы видим два спинорных индекса — верхний и нижний. Умножим эту матрицу на $(C)^{\alpha\xi}\left(C^{-1}\right)_{\xi\mu}$ справа.

$$(\gamma^a)^{\beta}_{\alpha}(C)^{\alpha\xi}\left(C^{-1}\right)_{\xi\mu}=i(\gamma^a)^{\beta}_{\alpha}\left(\gamma^0\gamma^2\right)^{\xi}_{\alpha}\left(C^{-1}\right)_{\xi\mu}=i\left(\gamma^2\gamma^0\right)^{\alpha}_{\xi}(\gamma^a)^{\beta}_{\alpha}\left(C^{-1}\right)_{\xi\mu}=$$

$$=i\left(\gamma^2\gamma^0\gamma^a\right)^{\beta}_{\xi}\left(C^{-1}\right)_{\xi\mu}=i\Big(\left(\gamma^2\gamma^0\gamma^a\right)^T\Big)^{\xi}_{\beta}\left(C^{-1}\right)_{\xi\mu}=i\Big(\left(\gamma^2\gamma^0\gamma^a\right)^T C^{-1}\Big)_{\beta\mu}$$

Сам объект не изменился, но обрел запись с двумя нижними спинорными индексами. Аналогично получается запись с верхними индексами. Этот эффект можно назвать дизайном вида выражения.

Функции *RepresLowIndex* и *RepresUpperIndex* преобразуют данную запись в запись с опущенными (поднятыми) индексами. Индексы, подлежащие перемещению, должны быть указаны во втором аргументе w как множества.

Примеры:

$[> RepresUpperIndex(\Omega(\gamma_1,\gamma_2)_{a,b}*\Omega(\gamma_0,\gamma_3)_{b,c}*\Omega(\gamma_1,\gamma_3)_{u,d},\{a,d,u\});$

$$\Omega(\gamma_2,\gamma_1)_{b,a}\Omega(\gamma_0,\gamma_3)_{b,c}\Omega(C,\gamma_5)_{u,d}$$

$[> RepresLowIndex(gammaExpand(\Omega(\gamma_1,\gamma_2)_{a,b}*\Omega(\gamma_0,\gamma_3)_{b,c}*\Omega(\gamma_1,\gamma_3)_{c,d}),\{a,d\})$

$$\Omega(C,\gamma_5)_{u,d}I\Omega(C1)_{a,d}$$

7.7. SeriesByGrassman

SeriesByGrassman (A)

Эту и две последущие функции также можно отнести к области дизайна выражений. Функция *SeriesByGrassman* принимает грассманово выражение и группирует в нём слагаемые по степеням грассмановых переменных.

Примеры:

$[> SeriesByGrassman(G(x,y)+\Omega(\gamma_1,\gamma_2)_{a,b}*\theta_a+\Omega(\gamma_0,\gamma_3)_{b,c}*\Omega(\gamma_1,\gamma_3)_{u,d}*\theta_b*\theta_u-\Omega(\gamma_1)_{a,b}*\theta_a);$

$$G(x,y)+(\Omega(\gamma_1,\gamma_2)_{a,b}-\Omega(\gamma_1)_{a,b})*\theta_a+\Omega(\gamma_0,\gamma_3)_{b,c}*\Omega(\gamma_1,\gamma_3)_{u,d}*\theta_b*\theta_u$$

7.8. SplitByGrassman

SplitByGrassman (A)

Функция *SplitByGrassman* принимает грассманово выражение и выдаёт массив, элементами которого являются однородные по степеням грассмановых переменных части исходного выражения. Так в 1-м элементе массива окажется часть выражения, не содержащая грассмановых переменных. В n-м элементе будут находиться слагаемые, имеющие порядок $n-1$ по грассмановым переменным.

7.9. SplitByGrassman1

$SplitByGrassman1\,(A)$

Это скоростная модификация предыдущей функции. $SplitByGrassman1$ не занимается вынесением за скобки общих множителей.

7.10. RepresGL

$RepresGL\,(A, Six)$

Функция RepresGL вычисляет $GL(4)$-представление выражения A. Она принимает два аргумента. Первый — само выражение, второй *Six-set of indexes* — массив его свободных индексов (записывается в квадратных скобках — [], $[\alpha]$, $[\alpha, \beta]$).

Выражение не должно содержать термины C, $C1$ и γ_5.

Примеры:

$[\,> eval\,(RepresGL\,(\Omega(\gamma_1)_{a,b}, [a, b]))\,;$

$$\begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

7.11. Некоторые нетривиальные примеры

7.11.1 Пример 1

Вот пример исследования известного [2] тождества:

$$\left(C^{-1}\right)_{\alpha\beta}(\gamma_5)_{\gamma}{}^{\delta} + \left(C^{-1}\right)_{\beta\gamma}(\gamma_5)_{\alpha}{}^{\delta} + \left(C^{-1}\right)_{\gamma\alpha}(\gamma_5)^{\delta}_{\beta} + \left(\gamma_5 C^{-1}\right)_{\alpha\beta}\delta^{\delta}_{\gamma} + \left(\gamma_5 C^{-1}\right)_{\beta\gamma}\delta^{\delta}_{\alpha} + \left(\gamma_5 C^{-1}\right)_{\gamma\alpha}\delta^{\delta}_{\beta} = 0\,:$$

$[\,> IC\,(a,b,c,d) := \Omega\,(\gamma_5, C1)_{b,c}\,\delta_{a,d} + \Omega\,(\gamma_5, C1)_{c,a}\,\delta_{b,d} + \Omega\,(\gamma_5, C1)_{a,b}\,\delta_{c,d} + \Omega\,(C1)_{a,b}\,\Omega\,(\gamma_5)_{c,d} + \Omega\,(C1)_{b,c}\,\Omega\,(\gamma_5)_{a,d} + \Omega\,(C1)_{c,a}\,\Omega\,(\gamma_5)_{b,d}\,:$

$[\,> RepresLowIndex\big(CorrectIndex\big(gammaExpand\big(LikeGammaMember\big(gammaExpand\big(IC\,(a,b,c,d)\,\theta_a\theta_b\theta_c\big)\big)\big), \alpha\,\beta\,\xi\big), \{\alpha, b, \beta, \xi\}\,\big);$

$$-3\,\theta_\alpha\theta_\beta\theta_\xi\left(\delta_{\alpha,d}\Omega\,(\gamma_5, C1)_{\xi,\beta} + \Omega\,(C1)_{\xi,\beta}\,\Omega\,(\gamma_5)_{\alpha,d}\right)$$

7.11.2 Пример 2

$[\,> A := iC1\,\Omega\,(\gamma_0,\gamma_1,\gamma_3)_{\beta,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\alpha,\nu8}\,\theta_{\eta8}\theta_{\nu8} + iC1\,\Omega\,(\gamma_0,\gamma_1,\gamma_3)_{\beta,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\alpha,\mu8}\,\theta_{\eta8}\theta_{\mu8} + 2\,C2\,\Omega\,(\gamma_2)_{\alpha,\beta}\,\Omega\,(\gamma_0,\gamma_2)_{\mu8,\nu8}\,\theta_{\mu8}\theta_{\nu8} - C2\,\Omega\,(\gamma_2)_{\beta,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\alpha,\nu8}\,\theta_{\eta8}\theta_{\nu8} - C2\,\Omega\,(\gamma_2)_{\beta,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\alpha,\mu8}\,\theta_{\eta8}\theta_{\mu8} + iC1\,\Omega\,(\gamma_0,\gamma_1,\gamma_3)_{\alpha,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\beta,\nu8}\,\theta_{\eta8}\theta_{\nu8} + iC1\,\Omega\,(\gamma_0,\gamma_1,\gamma_3)_{\alpha,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\beta,\mu8}\,\theta_{\eta8}\theta_{\mu8} - C2\,\Omega\,(\gamma_2)_{\alpha,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\beta,\nu8}\,\theta_{\eta8}\theta_{\nu8} - C2\,\Omega\,(\gamma_2)_{\alpha,\eta8}\,\Omega\,(\gamma_0,\gamma_2)_{\beta,\mu8}\,\theta_{\eta8}\theta_{\mu8}\,:$

$[\,> CorrectIndex\,(gammaExpand\,(LikeGammaMember\,(A))\,, [\mu\,\nu])\,;$

$$-2\theta_\mu\theta_\nu\big(iC1\Omega\,(\gamma_0,\gamma_1,\gamma_3)_{\beta,\nu}\,\Omega\,(\gamma_0,\gamma_2)_{\alpha,\mu} - iC1\,\Omega\,(\gamma_0,\gamma_1,\gamma_3)_{\alpha,\mu}\,\Omega\,(\gamma_0,\gamma_2)_{\beta,\nu} - C2\,\Omega\,(\gamma_0,\gamma_2)_{\alpha,\mu}\,\Omega\,(\gamma_2)_{\beta,\nu} + C2\,\Omega\,(\gamma_2)_{\alpha,\beta}\,\Omega\,(\gamma_0,\gamma_2)_{\nu,\mu} + C2\,\Omega\,(\gamma_0,\gamma_2)_{\beta,\nu}\,\Omega\,(\gamma_2)_{\alpha,\mu}\big)$$

7.11.3 Пример 3

Пример взятия сечения многовалентного тензора:

$[> eval\left(subs\left(\beta = 2, RepresGL\left(\Omega\left(\gamma_3\right)_{a,b}\Omega\left(\gamma_0,\gamma_2\right)_{\beta,\mu}\theta_\mu,[a,b]\right)\right)\right);$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ I\theta_4 & -I\theta_3 & I\theta_2 & -I\theta_1 \end{bmatrix}$$

8. Дополнительные возможности

8.1. SusyDiff

$SusyDiff\,(A, B, Index)$

Эта команда обобщает стандартную на случай грассмановых функций. Прежде всего: в рабочем скрипте должен быть объявлен массив $p = [t, x, y, z, \theta]$, в котором в соответствии с сигнатурой предствлены координатные переменные рабочего супермногообразия

$[> p := array(0..4, [t, x, y, z, \theta]) :$.

Обратим внимание на то, что здесь используется три аргумента. Здесь: A-выражение, которое нужно продифференцировать, B- число $0 \ldots 4$- индекс элемента массива p, представляющего переменную дифференцирования. Если $B = 4$, т.е. выполняется операция $\frac{\partial}{\partial\theta^\alpha}$, необходим 3-й аргумент $Index$. $Index$ — это символ индекса грассмановой переменной, того самого α. Грассмановы переменные, встречающиеся в выражениях A, имеют индексы, которые суть — немы. Поэтому индекс, соответствующей дифференцируемой переменной, подлежит замене на индекс при переменной дифференцирования, который указывается в 3-тем аргументе.

Пример:

$[> SusyDiff\left(f\left(t,x\right)\Omega\left(\gamma_0,\gamma_3\right)_{\mu\lambda}\theta_\mu, 1, \tau\right);$

$$\frac{\partial f\left(t,x\right)}{\partial x}\Omega\left(\gamma_0,\gamma_3\right)_{\mu\lambda}\theta_\mu$$

$[> SusyDiff\left(f\left(t,x\right)\Omega\left(\gamma_0,\gamma_3\right)_{\mu\lambda}\theta_\mu, 4, \tau\right);$

$$f\left(t,x\right)\Omega\left(\gamma_0,\gamma_3\right)_{\tau\lambda}$$

9. Уровень одночленов

9.1. SpecialMultiply

$SpecialMultiply\,(A, B)$

Процедура $SpecialMultiply$ — произведение полиномов A и B. Является подводящей процедурой для $gammaExpand$.

9.2. PutIndex

$PutIndex\,(A, w)$

Процедура $PutIndex$ заменяет спинорные индексы в мономе. Аргументы функции имеют тот же смысл, что и в $CorrectIndex$, но A - одночлен.

9.3. ConcretReprMnm

$ConcretReprMnm(A, Six)$

Является звеном цепочки для $RepresGL$. Вычисляет $GL(4)$-представление одночлена A. Аргументы функции имеют тот же смысл, что и в $RepresGL$, но A - одночлен.

9.4. Transps

Transps (A)

Процедура *Transps* принимает в качестве аргумента выражение мультипликативного вида $f\left(x^1,\ldots,x^n\right)\left(\gamma^{a^i_{m_1}}\gamma^{a^i_{m_2}}\ldots\gamma^{a^i_{m_i}}\right)^{\beta}_{\alpha}$ (в Maple–записи — $f\left(x_1,\ldots,x_n\right)\Omega\left(\gamma_{a_{m_1,i}}\gamma_{a_{m_2,i}}\ldots\gamma_{a_{m_i,i}}\right)_{\alpha\beta}$). Возвращает транспонированное выражение.

9.5. ComRule

ComRule (A,B)

Процедура ComRule выполняет произведение двух одночленов A и B. Она состоит из двух частей. Принимает в качестве аргумента два выражения вида $(\gamma^a,...,\gamma^c)^{\beta}_{\alpha}$ (в Maple–записи — $\Omega(\gamma_a,...,\gamma_c)_{\alpha,\beta}$). Сначала распознает порядок свертки произведений $\gamma^a\cdots\gamma^c$, затем производит выстраивание $\Omega(\gamma_a,...,\gamma_c)_{\alpha,\beta}$ по алфавитному возрастанию спинорных индексов. В случае образования свертки $\Omega(\gamma_a,...,\gamma_c)_{\alpha,\alpha}$, в силу бесследовости γ_a, возвращает 0.

9.6. MulGm

MulGm (A,B)

Процедура MulGm выполняет произведение двух одночленов A и B, принимает в качестве аргумента два выражения мультипликативного вида, содержащие несколько множителей $\Omega(\gamma_a,...,\gamma_c)_{\alpha,\beta}$. Возвращает упрощенное выражение.

10. Клиффордов уровень

Здесь собраны процедуры, выполняющие вычисления в алгебре Клиффорда. В нашем случае это алгебра матриц Дирака

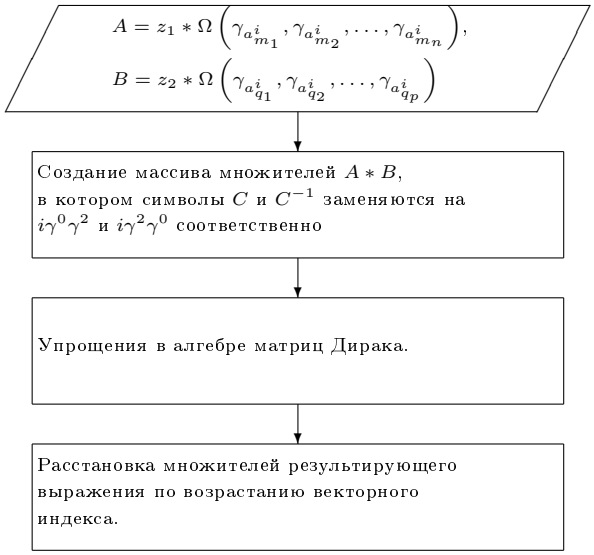
$$\gamma^a\gamma^b+\gamma^b\gamma^a=2\eta^{ab}\delta.$$

10.1. ComGm

ComGm (A,B)

Процедура ComGm выполняет произведение двух одночленов A и B. Она состоит из двух частей. Она принимает в качестве аргумента два выражения мультипликативного вида, содержащие $\Omega(\gamma_a,...,\gamma_c)$. Сначала происходит преобразование произведения $\gamma^a\cdots\gamma^c$, затем выстраивание $\gamma^a\cdots\gamma^c$ по возрастанию индекса. При $A=1$, фактически срабатывает только вторая часть. Процедура ComGm реализует, собственно, вычисления в алгебре матриц Дирака. Этот алгоритм состоит из двух частей: сначала выполняются необходимые коммутационные соотношения, затем производится расстановка множителей результирующего выражения по возрастанию векторного индекса.

Принципиальная логика этой процедуры такова:



Пример:

$[> ComGm(\Omega(\gamma_0, \gamma_3), \Omega(\gamma_1, \gamma_3));$

$$-\eta\eta_{3,3}\Omega(\gamma_0, \gamma_1)$$

10.2. TransDgm

$TransDgm\,(A)$

Процедура $TransDgm$ реализует транспонирование аргумента. Область определения этой процедуры есть множество символов $\{\gamma_0, \gamma_1, \gamma_2, \gamma_3, C, C1\}$. Возвращает пару: знак, символ. Возвращает пару: знак, символ. Приведём её текст:

```
TransDgm:=proc (a)
if a = gamma[0] then return 1, gamma[0] end if;
if a = gamma[1] then return -1, gamma[1] end if;
if a = gamma[2] then return 1, gamma[2] end if;
if a = gamma[3] then return -1, gamma[3] end if;
if a = gamma[4] then return 1, gamma[4] end if;
if a = gamma[5] then return 1, gamma[5] end if;
if a = C then return -1, C end if;
if a = C1 then return -1, C1 end if;
return 1, a :

end:
```

Переписав её, можно получить комплекс для работы с другой алгеброй Клиффорда.

11. Ядро системы

Сюда входят утилиты, решающие технические задачии системы. Их существование облегчает читаемость вызывающих их алгоритмов, что ценно с точки зрения программиста.

11.1. TransformToArray

$TransformToArray\,(A)$

Процедура $TransformToArray$ принимает в качестве аргумента выражение мультипликативного вида. Возвращает массив, состоящий из множителей. В случае если множитель имеет вид: $\Omega(\gamma_a, ..., \gamma_c)$, он отображается в участок массива с элементами $\gamma_a, ..., \gamma_c$ в предписанном порядке.

11.2. TransformSumToArray

$TransformSumToArray\,(A)$

Процедура $TransformSumToArray$ — технологическая. Преобразует полином в массив слагаемых.

Две последние процедуры возвращают в качестве результата объект типа $rTable$, который имеет коллекцию собственных стандартных методов, описанных в справке **Maple**. В частности, чтобы узнать количество элементов в результате, нужно вызвать функцию $ArrayNumElems(\dots)$.

11.3. Contract

$Contract\,(A, B)$

Процедура $Contract$ реализует свертку: $\Omega(\gamma_a, ..., \gamma_c)_{\alpha,\mu}\delta_{\mu,\beta} = \Omega(\gamma_a, ..., \gamma_c)_{\alpha,\beta}$. Вызывается если один из сомножителей - $\delta_{\mu,\beta}$.

11.4. Indetification

$Indetification\,(A)$

— булева функция. Написана для сокращения кода. Возвращает $true$, если аргумент - $\Omega\,(\dots)_{\dots}$ или $\delta_{\dots}$-конструкция и $false$ в противном случае.

11.5. DispatcherOfFactors

$DispatcherOfFactors\,(A, B)$

В силу того, что здесь объекты со спинорными индексами имеют неоднородное написание $\Omega\,(\dots)_{\alpha,\beta}$ и $\delta_{\alpha,\beta}$, при рассмотрении произведений возникают четыре, обрабатываемых по-разному, ситуации:

$$\Omega\,(\dots)_{\dots,\dots}\Omega\,(\dots)_{\dots,\dots}, \quad \delta_{\dots,\dots}\delta_{\dots,\dots}, \quad \delta_{\dots,\dots}\Omega\,(\dots)_{\dots,\dots} \quad \Omega\,(\dots)_{\dots,\dots}\delta_{\dots,\dots}.$$

Этому посвящена процедура DispatcherOfFactors — переключатель между свертками различных типов.

Заключение

Здесь приведены лишь самые необходимые сведения о комплексе SUSY_Diracus. Сказанного достаточно для эффективного использования этих инструментов. Мы надеемся, что эта работа позволит в некоторых случаях ускорить процесс вычислений.

В заключении приведем простую и довольно востребованную программу, вычисляющую коммутатор двух супергенераторов супералгебры $so\,(4, 2)$, супергенератора семейства супервращений

$$M_{ab} = x_a\partial_b - x_b\partial_a + \frac{1}{2}\theta^\alpha(\gamma_{ab})^\beta_\alpha\partial_\beta$$

и супергенератора дилатаций

$$D = x^a\partial_a + \frac{1}{2}\theta^\alpha\partial_\alpha.$$

Выражение

$$[M_{ab}, D] = 0$$

известно [2]. Здесь мы используем его для того, чтобы убедиться, что выражения супергенераторов в программе набраны верно. В случае успеха программа ничего выводить не должна. Именно вопрос вывода решается в строке

$for \quad u \quad from \quad 1 \quad to \quad 5 \quad do : if \quad L[u] <> 0 \quad then \quad print(i, u, L[u]) : fi : od :$

Структура программы — очевидна:

- загрузка $SUSY\_Diracus$,
- инициализационные настройки,

- запись супергенераторов,
- подготовка их к вычислениям,
- расчет суперкоммутатора,
- обработка супервыражения,
- вывод.

$[> read \quad "SUSY\_Diracus.mpl"$
$[> \eta\eta := Array(0..5, 0..5, fill = 0) :$
$[> \eta\eta_{0,0} := 1 : \eta\eta_{1,1} := -1 : \eta\eta_{2,2} := -1 : \eta\eta_{3,3} := -1 : \eta\eta_{5,5} := 1 :$
$[> p := array(0..4, [t, x, y, z, theta]) :$
$[> dp := array(0..4, [dt, dx, dy, dz, dth]) :$
$[> M := array(0..3, 0..3) :$
$[> for \quad i \quad from \quad 0 \quad to \quad 3 \quad do :$
$for \quad j \quad from \quad 0 \quad to \quad 3 \quad do :$
$M_{i,j} := \eta\eta_{i,i} * p_i * dp_j - \eta\eta_{j,j} * p_j * dp_i + \frac{1}{4} * p_4[\alpha 1] * \eta\eta_{i,i} * \eta\eta_{j,j} * \left(\Omega\left(\gamma_i, \gamma_j\right)_{\alpha 1,\tau} - \Omega\left(\gamma_j, \gamma_i\right)_{\alpha 1,\tau}\right) :$
$od :$
$od :$
$[> DD := sum(p_k * dp_k, k = 0..3) + \frac{\theta_\mu * \delta_{\mu,\tau}}{2} * dp_4[\tau] :$
$[> GX1 := array(0..4); GX2 := array(0..4) :$
$[> a := 1; b := 2 :$ ♯эта строка выведена неслучайно. Она является точкой управления. В данном случае задан супергенератор $M_{1,2}$.
$[> Wk := M_{a,b} :$
$for \quad i \quad from \quad 0 \quad to \quad 4 \quad do :$
$if \quad i \quad < \quad 4 \quad then \quad GX1_i \quad := \quad simplify\left(diff(Wk, dp_i)\right) \quad : \quad else \quad GX1_i \quad := simplify\left(diff(Wk, dp_i[\tau])\right) :$
$fi :$
$od :$
$[> Wk := DD :$
$for \quad i \quad from \quad 0 \quad to \quad 4 \quad do :$
$if \quad i \quad < \quad 4 \quad then \quad GX2_i \quad := \quad simplify\left(diff(Wk, dp_i)\right) \quad : \quad else \quad GX1_i \quad := simplify\left(diff(Wk, dp_i[\tau])\right) :$
$fi :$
$od :$
$[> for \quad i \quad from \quad 0 \quad to \quad 4 \quad do :$
$s \quad := \quad add(subs(\tau \quad = \quad \kappa, GX1[k]) * SusyDiff(GX2[i], k, \kappa) - subs(\tau \quad = \quad \kappa, GX2[k]) * SusyDiff(GX1[i], k, \kappa), k = 0..4) :$
$SLy := CorrectIndex(SimplifyByTh(expand(s)), [\xi 1, \xi 2, \xi 3, \xi 4]) :$
$SLy := gammaExpand(SLy) :$
$SLy := ExamExpress(SLy) :$
$SLy := LikeGammaMember(SLy) :$
$L := SplitByGrassman(SLy) :$
$for \quad u \quad from \quad 1 \quad to \quad 5 \quad do : if \quad L[u] <> 0 \quad then \quad print(i, u, L[u]) : fi : od :$
$od :$
$[>$

Если оставить команду $print(i, u, L[u])$; в той же строке без всяких условий, получится вывод результата вычисления коммутатора

$$[M_{ab}, D]$$

по компанентам и по степеням грассмановых переменных.

Файлы этого и других примеров дуступны по адресу: www.miklul.ru/Maple.

Благодарности

На описанный здесь комплекс получено свидетельство о государственная регистрация программы для ЭВМ №2021665791. Электронный охранный документ доступен по ссылке: https://fips.ru/EGD/5ad1a247-d432-471e-8b91-629529ea6420/2021665791.eod.pdf

Список литературы

1. К 80-летию Виталия Александровича Ростовцева // ПРОГРАММИРОВАНИЕ. Издательство: Московский государственный университет им. М.В. Ломоносова, Российская академия наук, Российская академия наук (Москва) ISSN: 0132-3474. Том: 38, Номер: 3, 2012. С. 79–80.

2. Мочалов С.В. Эффекты теории струн и суперсимметрий в космологических и небесных объектах // Кандидатская диссертация – , – , 1994.

3. Peeters K. Introducing Cadabra: a symbolic computer algebra system for field theory problems 2007. https://arxiv.org/abs/hep-th/0701238v2

4. Ablamowicz R., Fauser B. Clifford and Grassmann Hopf algebras via the BIGEBRA package for Maple(R). https://arxiv.org/pdf/math-ph/0212032.pdf

5. Ablamowicz R. CLIFFORD*: A Maple 13 Program for Clifford Algebra Computing. https://www.math.tntech.edu/rafal/cliff13/files/Tutorial_AGACSE2008.pdf

6. Березин Ф.А. *Введение в алгебру и анализ с антикоммутирующими переменными.* М.: МГУ, 1983. 208 с.

7. Pyry Kuusela "GammaMaP" – A Mathematica Package for Clifford Algebras, Gamma Matrices and Spinors https://arxiv.org/pdf/1905.00429

8. Шифман М.А. Суперсимметрия для начинающих. МАТЕРИАЛЫ 3 ЗИМНЕЙ ШКОНЫ ЛИАФ УДК 530.1 изд-во ЛИЯФ 1986.

9. Aminova A.V., Lyulinsky M.Kh. Non-vanishing cosmological constant effect in super-Poincare-invariant Universe. https://arxiv.org/abs/1904.07156v1

10. Лихтман Е.П. Суперсимметрия — 30 лет тому назад. *УФН.* 2001. 171:9. С. 1025–1032.

11. Талышев А.А. *Reduce в задачах математической физики.* Учебное пособие. Версия от 10 сентября 2012 г.

12. Фаронов В. *Turbo Pascal 7.0 Начальный курс.* М.: Издательство "ОМД Групп", 2003. 616 с.

13. Вайсфельд М. *Объектно-ориентированный подход.* Издательский дом "Питер", 2019. 256 с.

14. Аминова А.В., Мочалов С.В. Суперпространство Минковского как инвариант супергруппы Пуанкаре. *Известия вузов. Математика.* 1994. № 3. С. 5–12.

15. Аминова А.В., Люлинский М.Х., Мочалов С.В. Сферически симметричное суперпространство // В кн. “Новейшие проблемы теории поля. 1998.” - Казань: Хэтер, 1999. С. 222–230.

References

1. To the 80th anniversary of Vitaly Alexandrovich Rostovtsev // PROGRAMMING. Publishing House: Lomonosov Moscow State University, Russian Academy of Sciences, Russian Academy of Sciences (Moscow) ISSN: 0132-3474. 2012, vol. 38, no. 3, pp. 79–80.

2. Mochalov S.V. Effects of string theory and supersymmetry in cosmological and celestial objects // PhD dissertation – , – , 1994.

3. Peeters K. Introducing Cadabra: a symbolic computer algebra system for field theory problems 2007. https://arxiv.org/abs/hep-th/0701238v2

4. Ablamowicz R., Fauser B. Clifford and Grassmann Hopf algebras via the BIGEBRA package for Maple(R). https://arxiv.org/pdf/math-ph/0212032.pdf

5. Ablamowicz R. CLIFFORD*: A Maple 13 Program for Clifford Algebra Computing. https://www.math.tntech.edu/rafal/cliff13/files/Tutorial_AGACSE2008.pdf

6. Berezin F.A. *Introduction to algebra and analysis with anticommuting variables.* Moscow: MSU, 1983. 208 p.

7. Pyry Kuusela "GammaMaP" – A Mathematica Package for Clifford Algebras, Gamma Matrices and Spinors https://arxiv.org/pdf/1905.00429

8. Shifman M.A. Supersymmetry for beginners. MATERIALS OF THE 3rd WINTER SCHOOL OF THE LNPI UDC 530.1 publishing of the LNPI, 1986.

9. Aminova A.V., Lyulinsky M.Kh. Non-vanishing cosmological constant effect in super-Poincare-invariant Universe. https://arxiv.org/abs/1904.07156v1

10. Lichtman E.P. Supersymmetry — 30 years ago. *UFN*. 2001. 171:9. pp. 1025–1032.

11. Talyshev A.A. *Reduce in mathematical physics problems.* Study guide. Version of September 10, 2012.

12. Faronov V. *Turbo Pascal 7.0 Initial course.* Moscow: Publishing house "OMD Group", 2003. 616 p.

13. Weisfeld M. *Object-oriented approach.* Publishing house "Peter", 2019. 256 p.

14. Aminova A.V., Mochalov S.V. Minkowski superspace as an invariant of the Poincare supergroup. *Russian Mathematics (Iz. VUZ) Math*, 1994, no. 3, pp. 5–12.

15. Aminova A.V., Lyulinsky M.Kh., Mochalov S.V. Spherically symmetric superspace // In the book. "The recent problems of field theory. 1998." - Kazan: Hater, 1999. Pp. 222–230.

Авторы

Люлинский Михаил Хаимович, кафедра теории относительности и гравитации, институт физики, Казанский федеральный университет, ул. Кремлевская, 16а, г. Казань, 420008, Россия.
E-mail: miklul@rambler.ru

Просьба ссылаться на эту статью следующим образом:
Люлинский М. Х. Программный комплекс для работы с матрицами дирака SUSY_DIRACUS. *Пространство, время и фундаментальные взаимодействия.* 2022. № 39. С. 54—69.

Authors

Lyulinsky Mikhail Khaimovich, Physics and Mathematics, Professor, Theory of Relativity and Gravity, Kazan Federal University, ul. Kremlyovskaya, 16a, Kazan, 420008, Russia.
E-mail: miklul@rambler.ru

Please cite this article in English as:
Lyulinsky M. Kh. Soft for Dirac's matrices manipulation — SUSY_DIRACUS package. *Space, Time and Fundamental Interactions*, 2022, no. 39, pp. 54–69.