ISSN 1810-0198. Вестник Тамбовского университета. Серия: естественные и технические науки

Том 24, № 126 2019

DOI 10.20310/1810-0198-2019-24-126-166-178
УДК 517.312, 004.421.6

Процедурная интерпретация алгоритмов символьного интегрирования в системе MathPartner

Вячеслав Алексеевич КОРАБЕЛЬНИКОВ
ФГБОУ ВО «Тамбовский государственный университет им. Г.Р. Державина»
392000, Российская Федерация, г. Тамбов, ул. Интернациональная, 33
ORCID: https://orcid.org/0000-0002-3920-8373, e-mail: korabelnikov.va@gmail.com

Procedural interpretation of symbolic integration algorithms in MathPartner system

Vyacheslav A. KORABELNIKOV
Derzhavin Tambov State University
33 Internatsionalnaya St., Tambov 392000, Russian Federation
ORCID: https://orcid.org/0000-0002-3920-8373, e-mail: korabelnikov.va@gmail.com

Аннотация. Работа посвящена разработке библиотеки процедур для системы компьютерной алгебры MathPartner. Разрабатывается программная реализация алгоритмов символьного интегрирования. Решение задачи символьного интегрирования разбивается на три этапа. На первом этапе подынтегральное выражение приводится к виду, необходимому для применения алгоритма Риша. Приведено описание соответствующих процедур, которые сводят подынтегральную функцию к выражению, содержащему конечный набор арифметических операций и композиций логарифмических функций и экспонент, а также формируют набор регулярных мономов. На втором этапе осуществляется интегрирование дробной части подынтегрального выражения. Дано описание процедур, позволяющих привести дробную часть к виду, необходимому для применения алгоритма интегрирования. На третьем этапе проводится интегрирование полиномиальной части подынтегрального выражения. Получены процедуры, позволяющие в зависимости от вида подынтегрального выражения применить соответствующие алгоритмы интегрирования. В приложении приводится описание команд языка пользователя системы MathPartner, которые предназначены для вычисления интегралов в символьном виде.

Ключевые слова: система компьютерной алгебры; система MathPartner; символьное интегрирование

Благодарности: Работа выполнена при поддержке РФФИ (проект № 16-07-00420). Автор благодарит профессора Г. И. Малашонка за постановку задачи и многочисленные обсуждения, а также выражает признательность участникам семинара по компьютерной алгебре ТГУ за критику.

Для цитирования: *Корабельников В.А.* Процедурная интерпретация алгоритмов символьного интегрирования в системе MathPartner // Вестник Тамбовского университета. Серия: естественные и технические науки. Тамбов, 2019. Т. 24. № 126. С. 166–178. DOI 10.20310/1810-0198-2019-24-126-166-178.

Abstract. The work is devoted to the development of a procedure library for the computer algebra system MathPartner. A software implementation of symbolic integration algorithms is being developed. The solution of the problem of symbolic integration is divided into three stages. At the first stage, the integrand is reduced to the form necessary for applying the Rish algorithm. A description is given of the corresponding procedures that reduce the integrand to an expression containing a finite set of arithmetic operations and compositions of logarithmic functions and exponentials, and also make a set of regular monomials. At the second stage, the integration of the fractional part of the integrand is performed. A description is given of the procedures that reduce the fractional part to the form required for the application of the integration algorithm. At the third stage, the polynomial part of the integrand is integrated. Procedures are obtained that allow, depending on the type of the integrand, to apply the appropriate integration algorithms. The appendix contains a description of the user's language commands of the MathPartner system, which are designed to calculate integrals in symbolic form.

Keywords: computer algebra system; MathPartner computer algebra system; symbolic integration

Acknowledgements: The work is partially supported by the Russian Foundation for Basic Research (project no. 16-07-00420).
The author thanks Professor G. I. Malashonok for posing the problem and numerous discussions, and also expresses gratitude to the participants of the TSU Computer Algebra Workshop for criticism.

For citation: Korabelnikov V.A. Protsedurnaya interpretatsiya algoritmov simvol'nogo integrirovaniya v sisteme MathPartner [Procedural interpretation of symbolic integration algorithms in MathPartner system]. *Vestnik Tambovskogo universiteta. Seriya: estestvennye i tekhnicheskie nauki – Tambov University Reports. Series: Natural and Technical Sciences*, 2019, vol. 24, no. 126, pp. 166–178. DOI 10.20310/1810-0198-2019-24-126-166-178. (In Russian, Abstr. in Engl.)

Введение

Система компьютерной алгебры — это программный комплекс, предназначенный для выполнения символьных и численных вычислений. Одной из современных систем компьютерной алгебры является MathPartner. В отличие от большинства известных систем, MathPartner представляет из себя web-сервис, который свободно доступен в Интернете. Язык этой системы, называемый Mathpar, является по сути некоторым диалектом языка LaTeX. MathPartner позволяет сохранять и текст задания, и промежуточные вычисления, и результат вычислений. Пользователь имеет возможность сохранять исходный текст, текст в формате LaTeX и в виде изображения (pdf, jpg) [1].

Система MathPartner содержит большую библиотеку процедур для символьно-численных вычислений. Одним из пакетов этой библиотеки является пакет процедур для символьного интегрирования. Алгоритмы, лежащие в основе этой библиотеки процедур,

описаны в [2]. В этой статье приведены теоретические основы алгоритмов символьного интегрирования, реализованные в системе MathPartner. В рамках этих алгоритмов строится набор регулярных мономов и подынтегральное выражение представляется в виде суммы правильной дроби и полинома. Показан на алгоритмическом уровне процесс интегрирования полинома и правильной дроби. Рассматривается структура пакета процедур для символьного интегрирования и даются характеристики основным процедурам. Приводится общая блок-схема всего алгоритма символьного интегрирования.

1. Формирование набора регулярных мономов

В данной статье используются обозначения, принятые в [3, с. 213–229].

Процедура `integrate`. В системе MathPartner для интегрирования функций разработана процедура `integrate`. На входе процедура получает подынтегральную функцию. Процедура `integrate` вызывает процедуру `toComplex`, которая подготавливает подынтегральное выражение к процессу интегрирования. Полученное выражение передается процедуре `mainProcOfInteg`, которая вычисляет первообразную функции, выраженную в виде комбинации логарифмов и экспонент. После получения первообразной процедура `integrate` производит ее упрощение. Комбинации логарифмов и экспонент заменяются на тригонометрические, обратные тригонометрические, гиперболические, обратные гиперболические функции. Логарифмы от произведений заменяются на соответствующие суммы логарифмов. Появившиеся числовые слагаемые удаляются из первообразной. Полученная функция выдается пользователю в качестве ответа.

Для применения алгоритма Риша подынтегральная функция должна быть выражена через комбинацию натуральных логарифмов и экспонент [4, с. 225]. Для преобразования подынтегральной функции процедура `integrate` вызывает процедуру `toComplex`.

Процедура `toComplex`. Реализованная в пакете обработки функций системы MathPartner, процедура `toComplex` предназначена для проведения замены тригонометрических, обратных тригонометрических, гиперболических, обратных гиперболических функций на соответствующие комбинации логарифмов и экспонент, и использует следующие известные формулы:

$$\sin(x) = (\exp(ix) - \exp(-ix))/2i, \qquad \cos(x) = (\exp(ix) + \exp(-ix))/2,$$

$$\operatorname{arctg}(x) = i/2(\ln(1 - ix) - \ln(1 + ix)), \qquad \operatorname{arcctg}(x) = i/2(\ln((x - i)/x) - \ln((x + i)/x)),$$

$$\operatorname{sh}(x) = (\exp(x) - \exp(-x))/2, \qquad \operatorname{ch}(x) = (\exp(x) + \exp(-x))/2,$$

$$\operatorname{arctgh}(x) = 1/2\ln((1 + x)/(1 - x)), \qquad \operatorname{arcctgh}(x) = 1/2\ln((x + 1)/(x - 1)).$$

Процедура `mainProcOfInteg`. На входе получает функцию, преобразованную процедурой `toComplex`. Возвращает первообразную от входного выражения.

Процедура `mainProcOfInteg` вызывает процедуру `convertLOGandPOWtoLNandEXP`, которая предназначена для замены показательных функций на соответствующие комбинации логарифмов и экспонент: $f(x)^{g(x)} = \exp(g(x)\ln(f(x)))$. Эта процедура также

приводит логарифмы и показательные функции к основанию e: $a^x = \exp(a\ln(x))$, $\log_a(x) = \ln(x)/\ln(a)$.

Процедура `mainProcOfInteg` вызывает процедуру `makeListOfLNandEXP`, которая составляет список логарифмов и экспонент, содержащихся в подынтегральном выражении. Список составляется таким образом, что чем глубже вложены в аргументах логарифм или экспонента, тем ближе к началу списка они находятся. Например, для подынтегрального выражения $\exp(x^2\ln(x^3+x\ln(x)+\exp(x^2)))$ список будет иметь вид:

$$[\ln(x), \exp(x^2), \ln(x^3+x\ln(x)+\exp(x^2)), \exp(x^2\ln(x^3+x\ln(x)+\exp(x^2)))].$$

Процедура `mainProcOfInteg` вызывает процедуру `makeRegularMonomialsSequence`, которая преобразует список логарифмов и экспонент в набор регулярных мономов. Происходит построение наименьшего поля, которому принадлежит подынтегральное выражение. Исходным является поле $C(x)$ рациональных функций над полем комплексных чисел (см. [4, с. 2254]). Это поле последовательно расширяется добавлением в него логарифмов и экспонент из списка.

Подынтегральная функция представляется в виде рациональной функции от последнего регулярного монома из набора. Если выражение представляет собой неправильную дробь, то выделяется целая часть, и дробь приводится в правильный вид. Процедура `mainProcOfInteg` вызывает процедуру `polPartInteg` для интегрирования полиномиальной части подынтегрального выражения и процедуру `fracPartInteg` для интегрирования дробной части подынтегрального выражения. Собирает полученные результаты и возвращает в процедуру `integrate`.

Процедура `makeRegularMonomialsSequence`. Действия процедуры `makeRegularMonomialsSequence` основаны на следующем алгоритме. Просматриваются все логарифмы в списке. Если аргумент логарифма представляет собой дробь $\xi = p/q$, то в подынтегральном выражении $\ln(\xi)$ заменяется на $\ln(p)-\ln(q)$. $\ln(\xi)$ меняется в списке на $\ln(p)$. $\ln(q)$ вставляется в список после $\ln(p)$.

Просматриваются последовательно все элементы списка логарифмов и экспонент начиная с первого элемента. Начальный элемент списка является регулярным мономом, т. е. расширяет поле $C(x)$. Этот элемент остается в списке. Далее для определения регулярных мономов используется следующий алгоритм.

1. Пусть рассматриваемым элементом списка является логарифм. Тогда происходит проверка, выражается ли рассматриваемый элемент списка через предыдущие. Пусть $\theta_1, \theta_2, \ldots, \theta_{k-1}$ — уже добавленные в $C(x)$ регулярные мономы, $\ln(f_k)$ — рассматриваемый элемент списка, E — множество индексов i, $0 \le i \le k-1$, таких что θ_i — экспонента, L — множество индексов j, $0 \le j \le k-1$, таких что θ_j — логарифм. Согласно структурной теореме [3, с. 225], если рассматриваемый логарифм принадлежит уже сформированному полю, то его аргумент должен выражаться в виде произведения

$$f_k = c\prod_{i\in E}\theta_i^{n_i} \times \prod_{j\in L} f_j^{m_j},$$

где $n_i, m_j \in Q$, f_j — аргумент θ_j, $c \in C$. Поэтому производится поочередное деление аргумента рассматриваемого логарифма на предыдущие элементы списка. Если элементом списка является экспонента, то производится деление на саму экспоненту. Если элементом списка является логарифм, то производится деление на аргумент этого логарифма. Если деление проходит без остатка, то рассматриваемый элемент списка удаляется. Если деление происходит с остатком, то как в подынтегральном выражении, так и в последующих элементах списка производится замена рассматриваемого логарифма на сумму логарифма от остатка и элемента деления. Производится замена рассматриваемого элемента списка на логарифм, аргументом которого является остаток от деления.

2. Пусть рассматриваемым элементом списка является экспонента. Тогда происходит проверка, выражается ли рассматриваемый элемент списка через предыдущие. Пусть $\theta_1, \theta_2, \ldots, \theta_{k-1}$ — уже добавленные в $C(x)$ регулярные мономы, $\exp(f_k)$ — рассматриваемый элемент списка, E — множество индексов i, $0 \le i \le k-1$, таких что θ_i — экспонента, L — множество индексов j, $0 \le j \le k-1$, таких что θ_j — логарифм. Согласно структурной теореме [3, с. 225], если рассматриваемая экспонента принадлежит уже сформированному полю, то ее аргумент должен выражаться в виде линейной комбинации

$$f_k = c + \sum_{i \in E} n_i f_i + \sum_{j \in L} m_j \theta_j \ ,$$

где $n_i, m_j \in Q$, f_i — аргумент θ_i, $c \in C$. Составляется система линейных уравнений. Если она имеет решение, то рассматриваемый элемент удаляется из списка. Это решение позволяет выразить рассматриваемый элемент списка через предыдущие. В подынтегральном выражении и в последующих элементах списка меняется рассматриваемый элемент на его выражение через предыдущие элементы списка. Если система не имеет решения, то рассматриваемый элемент оставляется в списке без изменений.

Таким образом, список логарифмов и экспонент преобразуется в набор регулярных мономов. Процедура `makeRegularMonomialsSequence` передает полученный набор регулярных мономов в процедуру `mainProcOfInteg`.

2. Интегрирование дробной части

Процедура `fracPartInteg`. Для подготовки дроби к интегрированию процедура `fracPartInteg` производит вызов процедур `FactorPol_SquareFree` и `partialFraction`.

Процедура `FactorPol_SquareFree`, реализованная в пакете полиномиальных вычислений системы MathPartner, производит разложение знаменателя на свободные от квадратов множители.

Процедура `partialFraction` производит разложение дроби на сумму простейших дробей с помощью метода неопределенных коэффициентов.

Процедура `fracPartInteg` производит интегрирование отдельно каждого слагаемого из суммы. Согласно принципу Лиувилля (см. [4, с. 226]) и леммам о разложении (см. [4, с. 227,232]), интеграл от каждого слагаемого выражается в виде суммы

дробной функции и логарифмов с постоянными коэффициентами.

Если показатель степени, в которую возведен знаменатель рассматриваемого слагаемого, больше единицы, то процедура `fracPartInteg` вызывает процедуры `Hermite` и `LogarithmicPart`. Процедура `Hermite` разделяет интеграл на сумму рациональной и логарифмической части и вычисляет рациональную часть интеграла. Процедура `LogarithmicPart` вычисляет логарифмическую часть интеграла.

Если показатель степени, в которую возведен знаменатель рассматриваемого слагаемого, равен единице, то полином, стоящий в знаменателе, свободен от квадратов, и рациональная часть интеграла равна нулю. Процедура `LogarithmicPart` вычисляет логарифмическую часть интеграла.

Процедура `partialFraction`. Действия процедуры `partialFraction` основываются на следующих рассуждениях (см. [5, с. 38]). Пусть p/q — полученная дробная часть, $q = \prod q_i^i$, где q_i - свободный от квадратов полином. Тогда

$$p/q = p_{1,1}/q_1 + p_{2,1}/q_2 + p_{2,2}/q_2^2 + \ldots + p_{k,1}/q_k + p_{k,2}/q_k^2 + \ldots + p_{k,k}/q_k^k \ ,$$

где $p_{i,j} = A_{i,j}$ при $q_i = x - a$, $p_{i,j} = A_{i,j}x + B_{i,j}$ при $q_i = x^2 + ax + b$, где $A_{i,j}, B_{i,j}$ — числа, полученные с помощью метода неопределенных коэффициентов.

Процедура `Hermite`. Действия процедуры `Hermite` основаны на алгоритме Эрмита (подробнее см. [4, с. 220, 230, 235]).

$$\int \frac{p_{i,j}}{q_i^j} = -\frac{p_{i,j}b}{(j-1)q_i^{j-1}} + \int \frac{(j-1)p_{i,j}a + (p_{i,j}b)'}{(j-1)q_i^{j-1}} \ ,$$

где $q_i a + q_i' b = 1$ и $j > 1$. В результате применения этой формулы степень j, в которую возведен знаменатель, уменьшилась. Таким образом, эта формула повторно применяется пока $j \neq 1$.

Процедура `LogarithmicPart`. Действия процедуры `LogarithmicPart` основаны на следующем алгоритме (см. [4, с. 222]):

1. Интегрируемая дробь обозначается u/v.

2. Если числитель и производная знаменателя являются числами, то формируется ответ: $u\ln(v)$. Ответ передается в процедуру `fracPartInteg`.

3. Если u/v — дробь, числитель и знаменатель которой являются полиномами от x, то вычисляется результант $Res = resultant_x(u - yv', v)$ — полином новой переменной y.

Если u/v — дробь, числитель и знаменатель которой являются полиномами от $\ln(\xi)$, то вычисляется результант $Res = resultant_{\ln(\xi)}(u - yv', v)$ — полином новой переменной y.

Если u/v — дробь, числитель и знаменатель которой являются полиномами от $\exp(\xi)$, то вычисляется результант $Res = resultant_{\exp(\xi)}(u - y(v' - \deg(v)v\xi'), v)$ — полином новой переменной y.

4. Процедура `solvePolynomEq`, реализованная в пакете полиномиальных вычислений системы MathPartner, используется для нахождения всех корней результанта Res в поле комплексных чисел. Просматриваются все найденные корни. Упрощается очередной

корень. Если корень содержит переменную интегрирования, то интеграл от исходной функции не выражается в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде.

5. Определяется $V(y) = \text{НОД}(u - yv', v)$, если u/v — дробь, числитель и знаменатель которой являются полиномами от x или от $\ln(\xi)$; $V(y) = \text{НОД}(u - y(v' - \deg(v)v\xi'), v)$, если u/v — дробь, числитель и знаменатель которой являются полиномами от $\exp(\xi)$. Корень результанта подставляется в $V(y)$. Полученное выражение упрощается. Первообразная от дроби p/q выражается в виде формулы: $c\ln(V(c))$, где c — рассматриваемый корень результанта. Полученный логарифм прибавляется к общему ответу.

3. Интегрирование полиномиальной части

Процедура `polPartInteg`. Интегрирование полиномиальной части выполняет процедура `polPartInteg`. Возможны три случая:

1. Если полиномиальная часть — полином от x, то вызывается процедура `integPol`. Ее действия основаны на формуле $\int \sum a_i x^i dx = \sum a_i x^{i+1}/(i+1)$.

2. Если полиномиальная часть — полином от $\ln(\xi)$, то вызывается процедура `integPolLn`.

3. Если полиномиальная часть — полином от $\exp(\xi)$, то вызывается процедура `integPolExp`.

Процедура `integPolLn`. Действия процедуры `integPolLn` основываются на следующих рассуждениях. Пусть P — полиномиальная часть подынтегрального выражения. Согласно принципу Лиувилля (см. [4, с. 226]) и лемме о разложении (см. [4, с. 227]), $\int P = v_0 + \sum_{i=0}^{d} c_i \log(v_i)$, где $c_i \in C$, $v_0, v_1, \ldots, v_d$ — функции составленные из логарифмов и экспонент, содержащихся в выражении P, и функция v_0 является полиномом от $\ln(\xi)$. Обозначим $v_0 = \sum_{j=0}^{m+1} t_i \ln(\xi)^i$, $P = \sum_{i=0}^{m} p_i \ln(\xi)^i$, где t_i, p_i, ξ — функции, составленные из логарифмов и экспонент, содержащихся в выражении P. Подставляя в это уравнение выражение для P и v_0, получим выражение

$$\sum_{i=0}^{m} p_i \ln(\xi)^i = \sum_{i=0}^{m+1} t_i' \ln(\xi)^i + \sum_{i=0}^{m} (i+1) t_{i+1} \ln(\xi)^i \frac{\xi'}{\xi} + \sum_{i=1}^{d} c_i \frac{v_i'}{v_i} .$$

Для нахождения неизвестных $t_i, i = m+1, \ldots, 0$ приравниваем коэффициенты при одинаковых степенях $\ln(\xi)$ и получаем систему дифференциальных уравнений

$$\left\{ \begin{array}{rcl} 0 & = & t_{m+1}' \\ p_m & = & t_m' + (m+1) t_{m+1} \frac{\xi'}{\xi} \\ p_{m-1} & = & t_{m-1}' + m t_m \frac{\xi'}{\xi} \\ & \ldots & \\ p_1 & = & t_1' + 2 t_2 \frac{\xi'}{\xi} \\ p_0 & = & t_0' + t_1 \frac{\xi'}{\xi} + \sum_{i=1}^{d} c_i \frac{v_i'}{v_i} . \end{array} \right. \tag{1}$$

В процедуре `integPolLn` выполняются следующие действия. Создается список B длиной $m+2$, в который записываются решения уравнений системы (1). Решением первого уравнения является константа, которую невозможно определить рассматривая только первое уравнение. Эта константа определяется при решении второго уравнения. Поэтому изначально в элемент списка $B[m+2]$ записывается значение 0. Дальнейшие действия повторяются для каждого уравнения системы (1). При помощи процедуры `mainProcOfInteg` вычисляются $A1 = \int p_m$, $A2 = \int((m+2)B[m+2]/ln(\xi)')$. Если процедура `mainProcOfInteg` не вернула значение $A1$ или $A2$, то и интеграл от подынтегральной функции не выражается в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде. Если процедура `mainProcOfInteg` вернула значения $A1$ и $A2$, то в элемент списка $B[m+1]$ записывается значение разности $A1 - A2$. Если в выражении элемента списка $B[m+1]$ есть функция $ln(\xi)$, то выделяется выражение h, на которое умножен $ln(\xi)$. Если h содержит переменную интегрирования, то интеграл от подынтегральной функции не выражается в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде. Если h не содержит переменную интегрирования, происходит сложение $-h/(m+2)$ и выражения элемента списка $B[m+2]$. Результат записывается в $B[m+2]$ вместо старого значения. Происходит вычитание $h \cdot \ln(\xi)$ из выражения элемента списка $B[m+1]$, и результат записывается в элемент списка $B[m+1]$ вместо старого значения. Таким образом, найдено решение очередного уравнения.

Собирается ответ: $\sum_{i=0}^{m+2} B[i] \cdot \ln(\xi)^i$.

Процедура `integPolExp`. Действия процедуры `integPolExp` основываются на следующих рассуждениях. Пусть P — полиномиальная часть подынтегрального выражения. Согласно принципу Лиувилля (см. [4, с. 226]) и лемме о разложении (см. [4, с. 227]), $\int P = v_0 + \sum_{j=1}^{d} c_j \log(v_j)$, где $c_j \in C$, $v_0, v_1, \ldots, v_d$ — функции, составленные из логарифмов и экспонент, содержащихся в выражении P, и функция v_0 является полиномом от $\exp(\xi)$. Обозначим $v_0 = \sum_{i=-k}^{m} t_i \exp(\xi)^i$, $P = \sum_{i=-k}^{m} p_i \exp(\xi)^i$, где t_i, p_i, ξ — функции, составленные из логарифмов и экспонент, содержащихся в выражении P. Подставляя в это уравнение выражение для P и v_0, получим выражение

$$\sum_{i=-k}^{l} p_i \exp(\xi)^i = \sum_{i=-k}^{l} t_i' \exp(\xi)^i + \sum_{i=-k}^{l} i t_i \exp(\xi)^i u' + \sum_{j=1}^{d} c_j \frac{v_j'}{v_j} .$$

Приравниваем коэффициенты при одинаковых степенях $\exp(\xi)$ и получаем систему дифференциальных уравнений

$$\begin{cases} p_i &= t_i' + i t_i \xi', i \neq 0 \\ p_0 &= t_0' + \sum\limits_{j=1}^{d} c_j \frac{v_j'}{v_j} . \end{cases} \tag{2}$$

В процедуре `integPolExp` выполняются следующие действия. Создается список B длиной $m+k+1$, в который будут записываться решения системы дифференциальных

уравнений (2). В $B[i], i \neq 0$, записываются значения t_i. Рассматривается уравнение $p_i = t_i' + i \cdot \xi' \cdot t_i$, $i \neq 0$. Вызывается процедура `notDenom`, которая преобразует полученное уравнение к полиномиальному виду. Вызывается процедура `solveRischDiffEq`, которая решает дифференциальное уравнение и записывает решение в $B[i]$.

Для $i = 0$ вычисляется $\int p_0$ при помощи процедуры `mainProcOfInteg`, и результат записывается в $B[0]$.

Собирается ответ: $\sum_{i=-k}^{m} B[i] \exp(\xi)^i$.

Процедура `notDenom`. Действия процедуры `notDenom` основываются на следующих рассуждениях (см. [6]). Рассматривается дифференциальное уравнение $p_i = t_i' + i \cdot \xi' \cdot t_i$.

Пусть p_i — полином. Тогда t_i — тоже полином. Если ξ' — полином, то уравнение остается без изменений, а если $\xi' = v/w$, то чтобы «избавиться от знаменателей», уравнение домножается на w.

Пусть $p_i = r/g$. Тогда t_i имеет вид $t_i = a/b$. Если ξ' — полином, то получается уравнение

$$\frac{r}{g} = \frac{a'b - ab'}{b^2} + \frac{i\xi' a}{b} .$$

Следовательно, $b = \sqrt{g}$, $r = a'b - ab' + iu'ab = ba' + (-b' + iu'b)a$.

Если $\xi' = v/w$, то из соотношения

$$\frac{r}{g} = \frac{a'b - ab'}{b^2} + \frac{iva}{bw}$$

получается: $b = \sqrt{g/w}$, $r = a'bw - ab'w + ivab = bwa' + (ivb - b'w)a$.

Получено уравнение вида:

$$P = c_1 a' + c_2 a ,$$

где P, c_1, c_2, a — полиномы.

Процедура `solveRischDiffEq`. Действия процедуры `solveRischDiffEq` основаны на следующем алгоритме (см. [6]). Рассматривается дифференциальное уравнение относительно a: $P = c_1 a' + c_2 a$, где P, c_1, c_2 — полиномы либо от x, либо от $\ln(\psi)$, либо от $\exp(\psi)$. Возможны следующие три случая:

1. Если P, c_1, c_2 — полиномы от x, то неизвестная a — полином от x. Степень полинома a обозначается через n. Вычисляется $n = deg(P) - max\{deg(c_1 - 1, deg(c_2)\}$. Если $n < 0$, то исходное подынтегральное выражение не интегрируется в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде. Если $n = 0$, то $a = P/c_2$. Если $n > 0$, то приравниваются коэффициенты при одинаковых степенях x, составляется и решается система линейных уравнений. Решения уравнений являются коэффициентами неизвестного полинома. Если система линейных уравнений не имеет решения, то исходное подынтегральное выражение не интегрируется в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде.

2. Если P, c_1, c_2 — полиномы от $\ln(\psi)$, то неизвестная a — полином от $\ln(\psi)$. Степень полинома a обозначается через n. Вычисляется $n = deg(P) - max\{deg(c_1), deg(c_2)\}$. Если $n < 0$, то исходное подынтегральное выражение не интегрируется в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде. Если $n \geq 0$, то уравнение принимает вид:

$$P = c_1 \left(\sum_{i=0}^{n} a_i' \ln(\psi)^i + \sum_{i=1}^{n} i a_i \frac{\psi'}{\psi} \ln(\psi)^{i-1} \right) + c_2 \sum_{i=0}^{h} a_i \ln(\psi)^i.$$

Приравниваются коэффициенты при одинаковых степенях $\ln(\psi)$, и получается система дифференциальных уравнений. Пусть $a = \sum_{i=0}^{n} \alpha_i \ln(\psi)^i$, $c_1 = \sum_{i=0}^{m} \beta_i \ln(\psi)^i$, $c_2 = \sum_{i=0}^{s} \gamma_i \ln(\psi)^i$, $P = \sum_{i=0}^{t} p_i \ln(\psi)^i$. Возможны следующие случаи:

2.1. Степени полиномов c_1 и c_2 равны. Получается система дифференциальных уравнений вида:

$$\begin{cases} p_t &= \beta_m \alpha_n' + \gamma_m \alpha_n \\ p_{t-1} &= \beta_m \alpha_{n-1}' + \beta_{m-1} \alpha_n' + n \beta_m \alpha_n \frac{\psi'}{\psi} + \gamma_m \alpha_{n-1} + \gamma_{m-1} \alpha_n \\ & ... \end{cases}$$

Для решения каждого уравнения вызывается процедура `solveRischDiffEq`. Если очередное уравнение не имеет решения, то исходное подынтегральное выражение не интегрируется в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде.

2.2. Степень полинома c_1 больше степени полинома c_2. Получается система дифференциальных уравнений вида:

$$\begin{cases} p_t &= \beta_m \alpha_n' \\ p_{t-1} &= \beta_m \alpha_{n-1}' + \beta_{m-1} \alpha_n' + n \beta_m \alpha_n \frac{\psi'}{\psi} \\ & ... \\ p_{t-r} &= \beta_m \alpha_{n-r}' + ... + \beta_{m-r} \alpha_n' + n \beta_m \alpha_{n-r+1} \frac{\psi'}{\psi} + ... + \gamma_s \alpha_n \\ & ... \end{cases}$$

Для решения каждого уравнения вызывается процедура `mainProcOfInteg`. Если решение очередного уравнения не выражается в элементарном виде, то исходное подынтегральное выражение не интегрируется в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде.

2.3. Степень полинома c_1 меньше степени полинома c_2. Получается система уравнений вида:

$$\begin{cases} p_t &= \gamma_s \alpha_n \\ p_{t-1} &= \gamma_s \alpha_{n-1} + \gamma_{s-1} \alpha_n \\ & ... \\ p_{t-r} &= \beta_m \alpha_n' + \gamma_s \alpha_{n-r} + ... + \gamma_{s-r} \alpha_n \\ & ... \end{cases}$$

Для решения каждого уравнения выражается α_i.

3. Если P, c_1, c_2 — полиномы от $\exp(\psi)$, то неизвестная a — полином от $\exp(\psi)$. Наибольшая степень полинома a обозначается через n_2. Наименьшая отрицательная степень полинома a обозначается через n_1. Пусть степени P меняются от $-t_1$ до t_2, степени c_1 меняются от $-m_1$ до m_2, степени c_2 меняются от $-s_1$ до s_2. Тогда $n_2 = t_2 - max\{m_2, s_2\}$, $n_1 = t_1 - max\{m_1, s_1\}$. Уравнение принимает вид:

$$P = c_1 \sum_{i=-n_1}^{n_2} (\alpha_i' + i\alpha_i u') \exp(\psi)^i + c_2 \sum_{i=-n_1}^{n_2} \alpha_i \exp(\psi)^i.$$

Приравниваются коэффициенты при одинаковых степенях $\exp(\psi)$, и получается система дифференциальных уравнений. Пусть $a = \sum_{i=-n_1}^{n_2} \alpha_i \exp(\psi)^i$, $c_1 = \sum_{i=-m_1}^{m_2} \beta_i \exp(\psi)^i$, $c_2 = \sum_{i=-s_1}^{s_2} \gamma_i \exp(\psi)^i$, $P = \sum_{i=-t_1}^{t_2} p_i \exp(\psi)^i$. Возможны следующие случаи:

3.1. Наибольшая степень полинома c_1 больше либо равна наибольшей степени полинома c_2. Получается система дифференциальных уравнений вида:

$$\begin{cases} p_{t_2} = \beta_{m_2}(\alpha_{n_2}' + n_2\alpha_{n_2}\psi') \\ \quad ... \\ p_{t_2-r} = \beta_{m_2-r}(\alpha_{n_2}' + n_2\alpha_{n_2}\psi') + ... + \beta_{m_2}(\alpha_{n_2-r}' + (n_2-r)\alpha_{n_2-r}\psi') + \gamma_{s_2}\alpha_{n_2} \\ \quad ... \end{cases}$$

Для решения каждого уравнения вызывается процедура `solveRischDiffEq`. Если очередное уравнение не имеет решения, то исходное подынтегральное выражение не интегрируется в элементарном виде. Вычисление интеграла останавливается, и пользователь информируется о том, что функция не интегрируема в элементарном виде.

3.2. Наибольшая степень полинома c_1 меньше наибольшей степени полинома c_2 Получается система уравнений вида:

$$\begin{cases} p_{t_2} = \gamma_{s_2}\alpha_{n_2} \\ \quad ... \\ p_{t_2-r} = \beta_{m_2}(\alpha_{n_2}' + n_2\alpha_{n_2}\psi') + \gamma_{s_2-r}\alpha_{n_s} + ... + \gamma_{s_2}\alpha_{n_2-r} \\ \quad ... \end{cases}$$

Для решения каждого уравнения выражается α_i.

4. Заключение

Приведенная реализация алгоритмов символьного интегрирования не является полной. Необходимо провести в дальнейшем доработку некоторых процедур. В процедуре `makeRegularMonomialsSequence` требуется реализовать поддержку дробных степеней при определении трансцендентности логарифма (см. [3, с. 225]). В текущей версии алгоритма допускаются только целые степени.

Для устранения случаев, когда рассматриваемый элемент из списка регулярных мономов неоднозначно выражается через предыдущие, необходимо список перестроить.

Для этого надо разработать алгоритм такого перестроения списка логарифмов и экспонент (см. [3, с. 235]). Для каждого нового варианта списка надо организовать повторный запуск процедуры `makeRegularMonomialsSequence`. В текущей версии алгоритма такие случаи считаются неинтегрируемыми.

В процедуре `solveRischDiffEq` надо усовершенствовать алгоритм оценки степени полинома (см. [3, с. 240]).

В процедурах `partialFraction` и `notDenom` реализованы неполные и упрощенные алгоритмы, поэтому необходимо произвести в процедуре `partialFraction` замену алгоритма разбиения правильной дроби в сумму простых дробей алгоритмом для общего случая (см. [4, с. 273]), а в процедуре `notDenom` произвести замену алгоритма избавления от знаменателей алгоритмом для общего случая (см. [3, с. 237]).

Приложение. Как вычислять интегралы с помощью web-сервиса MathPartner

Чтобы найти первообразную, нужно выполнить следующие шаги:

1. Зайти на web-сайт MathPartner http://mathpar.cloud.unihub.ru/
2. Нажать на кнопку «Тетрадь».
3. В появившемся поле ввода текста выбрать числовое поле и название переменной с помощью оператора: `SPACE=Q[x]`.
4. После задания кольца нужно ввести оператор интегрирования `\int()`, в аргументе которого указать интегрируемую функцию. Затем нужно указать переменную интегрирования. Например, `\int( \sin(x) ) dx`.
5. Нажать кнопку «выполнить».

Например, чтобы найти первообразную от функции $\sin(x)$, нужно ввести следующий текст:

```
SPACE=Q[x];
\int( \sin(x) ) dx;
```

При нажатии на кнопку «выполнить» получим следующий ответ: `(-1) \cos(x)`.

В случае, если первообразная от функции не выражается в элементарном виде, то программа выдает первоначально заданный текст команды интегрирования. Например:

```
SPACE=Q[x];
\int( \exp(x^2) ) dx;
```

При нажатии на кнопку «выполнить» получим следующий ответ: `\int(\exp(x^2) )dx`.

Список литературы

[1] Г. И. Малашонок, *Руководство по языку "Mathpar"*, Издательство Тамбовского университета, Тамбов, 2013.

[2] В. А. Корабельников, "Алгоритмы символьного интегрирования в системе MathPartner", *Вестник Тамбовского университета. Серия: естественные и технические науки*, **24**:125 (2019), 75–89 DOI: 10.20310/1810-0198-2019-24-125-75-89.

[3] Е. В. Панкратьев, *Элементы компьютерной алгебры*, МГУ, М., 2007.

[4] Дж. Дэвенпорт, И. Сирэ, Э. Турнье, *Компьютерная алгебра. Системы и алгоритмы алгебраических вычислений*, МГУ, М., 1991.

[5] Г. М. Фихтенгольц, *Курс дифференциального и интегрального исчисления*. Т. 2, ФИЗМАТЛИТ, М., 2001.

[6] С. М. Тарарова, "К проблеме построения алгоритма символьного интегрирования", *Вестник Тамбовского университета. Серия: естественные и технические науки*, **17**:2 (2012), 607-617.

References

[1] G. I. Malaschonok, *Language guide "Mathpar"*, Publishing House of Tambov University, Tambov, 2013 (In Russian).

[2] V. A. Korabelnikov, "Symbolic integration algorithms in CAS MathPartner", *Tambov University Reports. Series: Natural and Technical Sciences*, **24**:125 (2019), 75–89 DOI: 10.20310/1810-0198-2019-24-125-75-89 (In Russian).

[3] E. V. Pankrat'yev, *Elements of computer algebra*, MSU, Moscow, 2007 (In Russian).

[4] J. Davenport, Y. Siret, E. Tournier, *Computer algebra. Systems and algorithms of algebraic computation*, Mir, Moscow, 1991 (In Russian).

[5] G. M. Fichtenholz, *Differential and Integral Calculus*. V. 2, PHYSMATLIT, Moscow, 2001 (In Russian).

[6] S. M. Tararova, "To the problem of constructing an algorithm for symbolic integration", *Tambov University Reports. Series: Natural and Technical Sciences*, **17**:2 (2012), 607-617 (In Russian).

Информация об авторе

Корабельников Вячеслав Алексеевич, аспирант, кафедра функционального анализа. Тамбовский государственный университет им. Г.Р. Державина, г. Тамбов, Российская Федерация. E-mail: korabelnikov.va@gmail.com
ORCID: https://orcid.org/0000-0002-3920-8373

Поступила в редакцию 20.02.2019 г.
Поступила после рецензирования 19.04.2019 г.
Принята к публикации 20.05.2019 г.

Information about the author

Vyacheslav A. Korabelnikov, Post-Graduate Student, Functional Analysis Department. Derzhavin Tambov State University, Tambov, the Russian Federation. E-mail: korabelnikov.va@gmail.com
ORCID: https://orcid.org/0000-0002-3920-8373

Received 20 February 2019
Reviewed 19 April 2019
Accepted for press 20 May 2019