### Оглавление

Основные термины и определения……………………………………………………………………2

Введение………………………………………………………………………………………………...3

Обзор и сравнительный анализ источников и аналогов……………………………………………..4

Описание функциональных требований……………………………………………………………...5

Описание нефункциональных требований…………………………………………………………...6

Реализация………………………………………………………………………………………………7

Результаты и планы……………………………………………………………………………………..8

Список источников……………………………………………………………………………………...9

Приложение…………………………………………………………………………………………….10

### Основные термины и определения

1. SymPy – библиотека языка Python с открытым исходным кодом для символьных вычислений.
2. Замена переменных в интеграле – такая замена, при которой x меняется на ɸ(t), а dx меняется на ɸ`(t)dt
3. Универсальная тригонометрическая подстановка – замена в интеграле вида. При такой подстановке , , ,
4. Тригонометрическая подстановка для интегралов вида
5. Тригонометрическая подстановка для интегралов вида
6. Тригонометрическая подстановка для интегралов вида

### Введение

В наше время компьютеризированные вычисления играют очень большую роль во всех областях жизни, в особенности в науке. Они используются в авиации, космонавтике, архитектуре. И если в 20-ом веке до развития вычислительных систем необходимые расчёты приходилось проводить руками, то сейчас для этого используются компьютеры.

Преимущества очевидны: компьютеры не допускают ошибок, им не требуется отдых, они могут перебрать тысячи способов решения задачи за короткий временной промежуток. Именно поэтому с развитием вычислительных мощностей и популяризацией компьютеров человечество перешло на компьютеризированные вычисления.

Однако, компьютеры всё ещё ограничены в том плане, что они могут проводить только алгоритмические вычисления. То есть им необходим алгоритм или программа, следуя которым они будут проводить расчёты. Поэтому, людям все ещё необходимо принимать участие для написания этих алгоритмов для компьютеров.

Так, например, библиотека символьных вычислений SymPy языка Python позволяет вычислять определённые и неопределённые интегралы, но с ограничениями. SymPy позволяет проводить замены переменных в интеграле, однако только такие, при которых между старой и новой переменной можно установить взаимно однозначное соответствие. Поэтому библиотека не позволяет считать интегралы с помощью тригонометрических подстановок, широко используемых в математическом анализе.

Цель данного проекта – реализовать метод (функцию класса) trig\_transform для проведения тригонометрических подстановок в определённом и неопределённом интеграле. Написать для него документацию и примеры использования по образцу, принятому в SymPy.

### Обзор и сравнительный анализ источников и аналогов

* **Метод transform – заменяет переменную в интеграле, пересчитывая пределы интегрирования (для определённых интегралов), используя встроенные функции библиотеки sympy**
  + Позволяет производить только такие замены переменных, при которых между новым и старым значением можно установить взаимноодноозначное соответствие.
  + Не позволяет провести замену в интеграле
* **Встроенный в библиотеку SymPy метод trigintegrate**
  + Позволяет вычислять интегралы от тригонометрических функций.
  + trigintegrate использует шаблоны для интегрирования, поэтому может вычислять только интегралы от конкретных функций и их линейных комбинаций, таких как, например
  + Поскольку используются шаблоны, trigintegrate не способен вычислить, например, интеграл

Но всё вышеописанное трудно назвать аналогами trig\_integrate, из-за отличий в заявленном функционале. На самом деле, аналогов trig\_integrate не существует, потому что SymPy единственная крупная библиотека для символьных вычислений в Python. Поэтому аналогами могут быть только функции из SymPy, перечисленные выше.

### Описание функциональных требований

Требования к функциональным характеристикам:

* Реализован метод trig\_transform класса Integral
  + trig\_transform принимает два аргумента – x (старая переменная), u (новая переменная)

x = f(u) или f(x) = u

* Дополнительно реализован (вспомогательный) метод \_calc\_limits для пересчёта пределов интегрирования в определенных интегралах

Требования к организации входных данных:

* Корректные переменные для замены (переменные – символы)
  + Поддерживаются функции вида:
    - u = a \* sin(x)
    - u = a \* cos(x)
    - u = a \* tg(x)
    - u = a / sin(x)

Требования к организации выходных данных:

* Интеграл с корректно проведённой заменой переменных
* Новые пределы интегрирования (если есть)
* Тот же самый интеграл, если замену провести нельзя
* Исключение ValueError в случае некорректного ввода или входных данных

Требования к временным характеристикам:

* Программа должна завершаться за разумное время
* Счётчик времени – если решение не найдено за разумное время (параметр по умолчанию time)

### Описание нефункциональных требований

* Требования к документации
  + doc\_string – на английском
    - Описание исключений и ошибок, которые могут возникать в trig\_transform
    - Примеры правильного и неправильного использования trig\_transform
  + Краткое описание метода trig\_transform и его возможностей
* Требования к примерам
  + Примеры применения замены переменной к типовым задачам из курса математического анализа
* Требования к тестам
  + Корректные тесты на каждую из тригонометрических подстановок
  + Тесты, некорректно использующие метод trig\_transform

При этом, при некорректных тестах метод trig\_transform должен сообщать об ошибке, а при обычных тестах выдавать правильный ответ.

* Дополнительные требования
  + Параметр debug, при котором пользователю выводится выражение, дифференциал новой переменной, и т.п.

### Реализация

Поскольку конечной целью проекта предполагается интегрирование метода **trig\_transform** в библиотеку символьных вычислений **SymPy** языка **Python,** метод реализован на языке Python3.8.

Краткое описание алгоритма работы метода **trig\_transform**:

* В качестве входных переменных метод принимает два выражения (встроенный в **SymPy** класс **Expression**): x, u, где x – заменяемое выражение, а u – заменяющее выражение, при этом выполняется один из пунктов ниже
  + x = f(u)
  + f(x) = u
* Проводятся необходимые проверки на корректность входных данных:
  + Исходная переменная не совпадает с переменной интегрирования
  + Число свободных переменных для замены больше одной
  + «Старая» переменная не является символом
  + «Новая» переменная не является символом
  + «Новая» переменная содержит несколько свободных переменных
* Получаем новый интеграл с заменёнными переменными, используя встроенную в **SymPy** функцию **subs**, заменяющую переменные в выражении
* Упрощаем полученный интеграл встроенной функцией **trigsimp**
* Для неопределённых интегралов возвращаем полученный интеграл в качестве результата
* Для определённых интегралов пересчитываем пределы интегрирования функцией **\_\_calc\_limits**
* Функция **\_\_calc\_limits**
  + Используя встроенную функцию **solve** решает уравнение u — f(x) = 0
  + Проверяет корректность пределов для подстановки u = a \* sin(x)
  + Пересчитывает пределы используя полученное функцией solve выражение u через x
* Возвращаем определенный интеграл с проведённой подстановкой

### Результаты и планы

Уже реализовано:

1. Основной код метода trig\_transform
2. Обработка исключений:
   * Исходная переменная не совпадает с переменной интегрирования
   * Число свободных переменных для замены больше одной
   * «Старая» переменная не является символом
   * «Новая» переменная не является символом
   * «Новая» переменная содержит несколько свободных переменных
3. Быстрая замена в случае, когда «Старая» переменная совпадает с «Новой» переменной

Необходимо реализовать:

1. Оптимизировать и отладить все компоненты
2. Написать документацию для метода trig\_transform
3. Написать документацию для вспомогательной функции \_ calc limits
4. Написать примеры использования для метода trig\_transform
5. Оформить примеры замены переменных из курса Математического Анализа
6. Написать docstring для интеграции кода в sympy
7. (опционально) добавить параметр debug для пошаговой замены переменных
8. (опционально) добавить параметр time для отсечки по времени замены

**Драфт проекта**: [**GitHub**](https://github.com/Axer1001/sympy-project/)

### Список источников

* ***Документация библиотеки SymPy*** – [[Электронный ресурс]](https://docs.sympy.org/)
* **Документация класса Integral** – [[Электронный ресурс]](https://docs.sympy.org/latest/modules/integrals/integrals.html?highlight=integral" \l "sympy.integrals.integrals.Integral)
* **Документация метода transform** – [[Электронный ресурс]](https://docs.sympy.org/latest/modules/integrals/integrals.html?highlight=sympy integrals transform" \l "sympy.integrals.integrals.Integral.transform)
* **Документация метода trigsimp** – [[Электронный ресурс]](https://docs.sympy.org/latest/modules/simplify/simplify.html?highlight=trigsimp" \l "module-sympy.simplify.trigsimp)
* ***Документация Python3.8*** – [[Электронный ресурс]](https://docs.python.org/3.8/)
* **Исходный код библиотеки SymPy** – [[Электронный ресурс]](https://github.com/sympy/sympy)
* **Основы** **математического** **анализа**. В 2-х ч. **Ильин** **В**.**А**., **Позняк** **Э**.**Г**. М.: Физматлит. Ч.2 - 2005, 7-е изд., 648с.

### Приложение

# ***Календарный план***

|  |  |
| --- | --- |
| Запланированные задачи | Срок выполнения |
| Изучение области исследования | 1 ноября — 20 декабря |
| Изучение SymPy | 11 – 20 января |
| Анализ существующих решений | 20 — 31 января |
| Написание КТ-1 | 1 февраля — 8 февраля |
| Написание метода trig\_transform:   * Возможные исключения метода trig\_transform * Метод trig\_transform | 24 февраля — 10 апреля |
| Написание КТ-2 | 10 апреля — 16 апреля |
| Правка КТ-2 | 16 апреля — 24 апреля |
| Написание документации по образцу, принятому в SymPy,  Написание примеров  Отладка компонентов | 24 апреля — 15 мая |