Клепиков Глеб, 325 группа, 2025

Отчет по программе

Цель работы

Целью данной работы является реализация программы, способной автоматически решать системы линейных уравнений в символьном виде. Программа должна:

1. принимать систему уравнений в инфиксной форме Lisp;
2. корректно обрабатывать произвольное количество уравнений и переменных;
3. находить:
    a. единственное решение;
    b. бесконечное множество решений (параметрическое представление);
    c. противоречивые системы;
4. выполнять символьные преобразования без использования матриц.

Общая идея алгоритма

В основе программы лежит рекурсивный алгоритм, реализованный в символьной форме. Вместо матриц используется представление уравнений в виде ассоциативных списков коэффициентов.

Каждое уравнение приводится к нормализованному виду:

a1*X1 + a2*X2 + ... + an*Xn + c = 0

Далее алгоритм последовательно:

1. выбирает ведущую переменную (pivot);
2. выражает её через остальные переменные;

3. подставляет полученное выражение в оставшиеся уравнения;
4. рекурсивно решает уменьшенную систему;
5. выполняет обратную подстановку;
6. анализирует тип решения.

Структура программы

Программа логически разбита на восемь основных этапов:

1. Парсер уравнений
2. Выражение переменной
3. Нормализация выражений
4. Подстановка в систему
5. Анализ уравнений
6. Рекурсивное решение системы
7. Обратная подстановка
8. Вывод результата

1. Парсер уравнений

Парсер преобразует входные уравнения, заданные в виде списков токенов, во внутреннее представление — список пар вида:

((X 2) (Y -1) (_const 3))

Основные функции парсера

- `is-number-token` — определяет, является ли токен числом;
- `is-variable-p` — проверяет, является ли токен переменной (символ с заглавной буквы);

- `token-to-number` — преобразует числовой токен в число;
- `parse-equation` — точка входа для парсинга одного уравнения;
- `parse-equation-direct` — рекурсивный разбор выражения;
- `add-coefficient-to-list` — добавление и объединение коэффициентов;
- `normalize-equation` — перенос правой части уравнения в левую;
- `clean-zero-coeffs` — удаление нулевых коэффициентов.

Результатом работы парсера является система уравнений в нормализованном виде.

2. Выражение переменной

На данном этапе из одного уравнения выражается выбранная переменная через остальные.

Пример:

2X + Y - 3 = 0 -> X = (3 - Y) / 2

Основные функции

- `express-variable` — интерфейс для выражения переменной;
- `collect-other-variables` — сбор остальных переменных;
- `build-symbolic-numerator` — формирование числителя;
- `express-variable-build-result` — построение итогового выражения.

Выражение представляется в префиксной форме Lisp.

3. Нормализация выражений

Перед подстановкой выражения в другие уравнения оно нормализуется.

Задачи этапа:

- приведение чисел к форме суммы;

- раскрытие деления суммы на число;
- унификация структуры выражений.

Основные функции

- `normalize-expr-for-substitution`;
- `normalize-division-simple`.

4. Подстановка в систему

Подстановка выражения вместо переменной производится во всех остальных уравнениях системы.

Алгоритм:

1. коэффициент переменной умножается на выражение;
2. каждый член суммы добавляется в уравнение;
3. исходная переменная удаляется из уравнения;
4. коэффициенты объединяются.

Основные функции

- `substitute-in-system`;
- `substitute-in-equation`;
- `add-sum-to-equation`;
- `add-term-to-equation`;
- `remove-variable`.

5. Анализ уравнений

На данном этапе проверяется специальный вид уравнений.

Проверки

- **Противоречие**:

0 = 5

- **Тождество**:

0 = 0

Основные функции

- `check-contradiction`;
- `check-tautology`;
- `find-pivot` — выбор ведущей переменной.

6. Рекурсивное решение системы

Основной алгоритм решения реализован в функции:

solve-system-recursive

Алгоритм:

1. если система пуста — вернуть накопленные решения;
2. если найдено противоречие — завершить;
3. если уравнение является тождеством — пропустить;
4. выбрать ведущую переменную;
5. выразить ее;
6. подставить в остальные уравнения;
7. вызвать рекурсию.

Таким образом система последовательно уменьшается.

7. Обратная подстановка

После получения выражений для всех переменных выполняется обратная подстановка.

Цель этапа:

- вычислить значения переменных;
- упростить выражения;
- получить числовые или параметрические решения.

Основные функции

- `back-substitute`;
- `evaluate-expression`;
- `simplify-expr`.

8. Вывод результатов

Программа корректно различает типы решений:

- **единственное решение**;
- **бесконечное множество решений**;
- **отсутствие решений**.

Также определяется множество свободных переменных.

Для вывода выражений используется преобразование из префиксной формы в инфиксную.

Основные функции

- `solve-system-print-solutions`;
- `solve-system-print-parametric`;
- `prefix-to-infix`.

Примеры работы

Программа корректно решает:

- системы с единственным решением;

  (solve-system '((2 * X + 2 * Y - Z = 3) (2 * X - Y + 3 * Z = -6) (3 * X + 2 * Z = -4)))

  "Решаем систему..."

  "Единственное решение:"

  (X = -2/3)

  (Z = -1)

  (Y = 5/3)

- недоопределённые системы;

  (solve-system '((X - Y - Z = 2)))

  "Решаем систему..."

  "Бесконечное количество решений:"

  (X = (2 + ((1 * Y) + (1 * Z))))

  (Y Z — СВОБОДНЫЕ ПЕРЕМЕННЫЕ)

- противоречивые системы;

  (solve-system

  '((X - Y = 1) (Y + Z = 2) (X + Z = 1)))