Отчет

Ногиной Софьи Олеговны, 324 группа, 2023 год

Описание проделанного варианта задания

Задание 9: Решение системы линейных уравнений методом исключений

Дана система M линейных уравнений с N неизвестными, N и M >= 1
Необходимо решить заданную систему линейных уравнений методом исключения переменных. Решение системы включает в себя следующие шаги преобразований уравнений:

В одном из уравнений некоторая переменная выражается через другие переменные Полученное для этой переменной выражение подставляется в остальные уравнения системы вместо всех вхождений указанной переменной

В результате выполнения этих шагов получается формула для вычисления значения одной переменной системы по значениям других, а также новая равносильная система уравнений, содержащая на одно уравнение и на одну переменную меньше, чем предыдущая. Далее шаги 1-2 повторяются для новой системы уравнений, и так продолжается до тех пор, пока не останется одно уравнение или пока не обнаружится противоречие.

Противоречие (неверное равенство) возникает, когда система не имеет решения, и в этом случае в качестве ответа должно быть выдано соответствующее сообщение. Если же в процессе преобразований одно из уравнений превратилось в тождество (например, 0=0), то исходная система уравнений была избыточна, и это уравнение можно исключить из решаемой системы.

В случае, когда в ходе преобразований системы получено одно уравнение с одной переменной, то из этого уравнения вычисляется значение этой переменной. Значения остальных переменных вычисляются по полученным в ходе преобразований формулам, и итоговое решение системы уравнений выводится в виде строк вида имя_переменной=значение.

Если же в результате преобразований остаётся одно уравнение, и оно содержит более одной переменной, то необходимо вывести на печать имена свободных переменных системы и формулы для вычисления остальных переменных по значениям свободных.

Исходная система уравнений задается как список строк, каждая строка соответствует уравнению и записывается как список входящих в уравнение компонент (знаков операций, чисел и переменных), которые разделяются пробелами.

Текст лисп-программы

```
; Функции для распарсивания системы уравнений
; Представление: ((X + 3Y + 2Z = 10)(Z -  5Q = 5))->(((X (1 1))(Y (3 1))(Z (2 1)))((5 1)(Z (1 1))(Q (-5 1))(10 1))

; Распарсивание
(defun parse(system)(cond
        ((null system) NIL)
        (T (cons (parse_pp (car system) 1) (parse (cdr system))))))
; Распарсивание полинома
(defun parse_pp(polinom sgn)(cond
        ((null polinom) NIL)
        ((or (eql (car polinom) '+) (eql (car polinom) '-)) (parse_pp_t polinom sgn))
        (T (parse_pp_t (cons '+ polinom) sgn))  ; Случай (<нет знака>X + 2Y = 10)))
(defun parse_pp_t(lst sgn)(cond
        ((null lst) NIL)
        ((eql (car lst) '+) (parse_op (cdr lst) sgn sgn))
        ((eql (car lst) '-) (parse_op (cdr lst) (* -1 sgn) sgn))
        ((eql (car lst) '=) (parse_pp (cdr lst) -1))))
(defun parse_op(lst num sgn)(cond
   ((null lst) (cons (make_koef (* -1 num)) NIL))
        ((numberp (car lst)) (parse_op (cdr lst) (* num (car lst)) sgn))
        ((eql (car lst) '*) (parse_op (cdr lst) num sgn))
        ((or (eql (car lst) '+) (eql (car lst) '-) (eql (car lst) '=)) (cons (make_koef (* -1 num)) (parse_pp_t lst sgn)))
        ((symbolp (car lst)) (cons (cons (car lst) (cons (make_koef num) NIL)) (parse_pp_t (cdr lst) sgn)))))

; Создание коэффициента из числа. Например, 1 -> (1 1)
```

```
(defun make_koef(num)(cons num (cons 1 NIL)))
; Проверка, что коэфициент нулевой. Например, (0 1)->T
(defun is_null(koef)(= 0 (car koef)))

; Функции для арифметических операций
; Функция для нахождения НОД по алгоритму Евклида
(defun nod(num1 num2)(cond
   ((< num1 0) (nod (* num1 -1) num2))
   ((= num1 num2) num1)
   ((> num1 num2) (nod (- num1 num2) num2))
   (T (nod num1 (- num2 num1)))))
; Функция для сокращения коэффициента. Например (10 5) -> (2 1)
(defun shorten(koef)(short (car koef) (cadr koef)))
(defun short(num1 num2)(cond
   ((= 0 num1)(cons 0 (cons 1 NIL)))
   (T (cons (/ num1 (nod num1 num2)) (cons (/ num2 (nod num1 num2)) NIL)))))
(defun plus(koef1 koef2)(plus_t (car koef1) (cadr koef1) (car koef2) (cadr koef2)))
(defun plus_t(p1 q1 p2 q2)(short (+ (* p1 q2) (* p2 q1)) (* q1 q2)))
(defun minus(koef1 koef2)(plus_t (car koef1) (cadr koef1) (* -1 (car koef2)) (cadr koef2)))
(defun mul(koef1 koef2)(mul_t (car koef1) (cadr koef1) (car koef2) (cadr koef2)))
(defun mul_t(p1 q1 p2 q2)(short (* p1 p2) (* q1 q2)))
(defun div(koef1 koef2)(mul koef1 (swap koef2)))
(defun swap(koef)(cond
   ((< (car koef) 0) (cons (* -1 (cadr koef)) (cons (* -1 (car koef)) NIL)))
   ((> (car koef) 0) (cons (cadr koef) (cons (car koef) NIL)))
   (T (print '(ошибка в SWAP)))))

; Метод спуска
; (car matrix) - список свободных вершин, (cadr matrix) - система уравнений.
; если нет решений, (cadr matrix) ~ T
(defun exclusion(matrix)(cond
   ((null (cadr matrix)) NIL) ; Не осталось уравнений (например, в процесе некоторые оказались излишними)
   ((eq T (cadr matrix)) (cons (cadr matrix) (cons (car matrix) NIL))); У системы нет решений
            ((null (cdadr matrix)) (end (car matrix) (caadr matrix)))
            ; последнее уравнение.(благодаря dev_level осталось уравнение с переменной/ыми). end проверяет, какого вида уравнение
осталось
            (T (normalize (caadr matrix) (exclusion (dev_level (car (caaadr matrix)) (my_step (caadr matrix) (cdadr matrix)) (car matrix)))))
            ; normalize - подставляет в уравнение вместо переменных числа, приводит подобные слагаемые.
))

; Функция, возвращающая матрицу с исключенным уравнением lst
(defun my_step(lst system)(cond
            ((null system) NIL)
            (T (cons (step_lst lst (car system) (in (car lst) (car system))) (my_step lst (cdr system))))))
; Функция для определения коэффициента. Например, (in '(X (10 1)) '((Y (10 1)) (X (5 1))))-> (1 2)
(defun in(el lst)(cond
            ((null lst) (make_koef 0))
            ((eq (car el) (caar lst)) (div (cadar lst) (cadr el)))
            (T (in el (cdr lst)))))
; Функция для исключения уравнения lst1 из уравнения lst2
(defun step_lst(lst1 lst2 koef)(cond
            ((= 0 (car koef)) lst2)
            ((null (cdr lst1)) (step_end lst2 (mul koef (car lst1)))) ; Дошли до последнего элемента в lst1
            (T (step_lst (cdr lst1) (step_el (car lst1) lst2 koef) koef))))
; Функция для исключения переменной
(defun step_el(el lst koef)(cond
   ((null (cdr lst))(cons (cons (car el) (cons (minus (make_koef 0) (mul (cadr el) koef)) NIL)) lst))
            ((eq (car el) (caar lst)) (cons (cons (car el) (cons (minus (cadar lst) (mul (cadr el) koef)) NIL)) (cdr lst)))
            (T (cons (car lst) (step_el el (cdr lst) koef)))))
(defun step_end(lst el)(cond
            ((null (cdr lst)) (cons (minus (car lst) el) NIL))
            (T (cons (car lst) (step_end (cdr lst) el)))))

; Дана вершина top, матрица matrix. Нужно исключить нулевые переменные
(defun dev_level(top matrix tops)(dev_lev top matrix (cons tops (cons NIL NIL))))
; (car ans) - список обнулившихся вершин, (cadr ans) - система уравнений без нулевых переменных
(defun dev_lev(top matrix ans)(cond
   ((eq (cadr ans) T) ans) ; У системы нет решений
   ((null matrix) (cons (not_free top (car ans)) (cdr ans)))
   (T (dev_lev top (cdr matrix) (dev_lst (car matrix) ans)))))
(defun dev_lst(lst ans)(dev_lst_t lst (car ans) (cadr ans) NIL))
(defun dev_lst_t(lst tops system res)(cond
   ((null (cdr lst)) (cons tops (cons (check_end res (car lst) system) NIL))) ; Не осталось переменных
   ((is_null (cadar lst)) (dev_lst_t (cdr lst) (add_top (caar lst) tops) system res))
   (T (dev_lst_t (cdr lst) tops system (cons (car lst) res)))))
(defun not_free(top lst)(cond
            ((null lst) NIL)
            ((eq (car lst) top) (cdr lst))
            (T (cons (car lst) (not_free top (cdr lst))))))
```

```
; функция, добавляющая вершину в список обнулившихся
(defun add_top(top tops)(cond
   ((member top tops) tops)
   (T (cons top tops))))
(defun check_end(beg_lst end_lst system)(cond
   ((and (null beg_lst) (is_null end_lst)) system) ; Избыточное уравнение
   ((null beg_lst) T) ; Нет решений
   (T (cons (append beg_lst (cons end_lst NIL)) system))))

; Функция, для обработки последнего уравнения:
; Добавляем в конец список свободных вершин
(defun end(tops lst)(cond
            ((null (cddr lst)) (cons (first_normalize lst) (cons (not_free (caar lst) tops) NIL)))  ; осталась одна переменная
            (T (cons (first_normalize lst) (cons (not_free (caar lst) (add_tops tops (cdr lst))) NIL)))  ; осталось несколько переменных
))
; Функция, нормализующая уравнение по первой переменной
(defun first_normalize(lst)(cons (caar lst) (f_norm (cdr lst) (cadar lst))))
(defun f_norm(lst koef)(cond
            ((null (cdr lst))(cons (div (car lst) koef) NIL)) ; последний элемент - часть без переменной
            (T (cons (cons (caar lst) (cons (minus (make_koef 0) (div (cadar lst) koef)) NIL)) (f_norm (cdr lst) koef)))))
; Функция для добавления списка вершин
(defun add_tops(tops lst)(cond
            ((null lst) tops)
            ((null (cdr lst)) tops) ; последняя переменная - без буквы
            (T (add_tops (add_top (caar lst) tops) (cdr lst)))))

; Функция для подстановки чисел из матрицы matrix в уранвение lst
; Нужно: 1) Если очередная вершина в lst есть в матрице matrix - подставляем ее значение, умноженное на коэффициент
; 2) Приводим подобные слагаемые 3) удаляем нулевые слагаемые 4) Выражаем первую вершину
(defun normalize(lst matrix)(cond
   ((eq (car matrix) T) matrix) ; У системы нет решений
   ((null matrix) (end (car matrix) lst))
   (T (cons (cons (caar lst) (not_null (similar (norm (cdr (first_normalize lst)) matrix NIL)))) matrix))))
; функция, подставляющая переменные matrix в уравнение lst
; not_null удаляет все нулевые переменные, similar - приводит подобные слагаемые (в случаях, когда есть свободные переменные)
(defun norm(lst matrix res)(cond
            ((null (cdr lst)) (cons (car lst) res)) ; переменные закончились
            (T (norm (cdr lst) matrix (subs (car lst) matrix res))) ; подстановка))
(defun subs(el matrix res)(cond
            ((and (null (cdr matrix)) (member (car el) (car matrix))) (cons el res)) ; матрица закончилась. Значит, в el - свободная переменная
            ((null (cdr matrix))(print 'ошибка)) ; ошибка
            ((eq (car el) (caar matrix)) (expr (cadr el) (cdar matrix) res))
            (T (subs el (cdr matrix) res))))
(defun expr(koef el res)(cond
            ((null (cdr el)) (cons (mul koef (car el)) res))
            (T (expr koef (cdr el) (cons (cons (caar el) (cons (mul koef (cadar el)) NIL)) res)))))
(defun similar(lst)(sim lst (make_koef 0) NIL))
(defun sim(lst num tops)(cond
            ((null lst) (cons num NIL))
            ((numberp (caar lst)) (sim (cdr lst) (plus num (car lst)) tops))
            ((member (caar lst) tops) (sim (cdr lst) num tops))
            (T (cons (sim_symb lst (caar lst) (make_koef 0)) (sim (cdr lst) num (cons (caar lst) tops))))))
; Функция для приведения подобных слагаемых для одной вершины
(defun sim_symb(lst top acc)(cond
            ((null lst) (cons top (cons acc NIL)))
            ((eql (caar lst) top) (sim_symb (cdr lst) top (plus acc (cadar lst))))
            (T (sim_symb (cdr lst) top acc))))
; Функция для исключения нулевых переменных
(defun not_null(lst)(cond
            ((null (cdr lst)) lst)
            ((= 0 (caadar lst)) (not_null (cdr lst)))
            (T (cons (car lst) (not_null (cdr lst))))))

; Главная функция
(defun main(system)(print(print_system(exclusion(not_null_system(sim_system(parse system)))))))
; Функция для приведения подобных слагаемых в системе
(defun sim_system(system)(cond
            ((null system) NIL)
            (T (cons (similar (car system)) (sim_system (cdr system))))
))
; Функция для удаления нулевых слагаемых из системы
(defun not_null_system(system)(dev_level NIL system NIL))

; Функция для результата. Возвращает список с тем, что нужно вывести
(defun print_system(ans)(cond
            ((or (null ans) (eq T (car ans))) "Система не имеет решений")
            ((null (cdr ans)) (cons (print_free (car ans)) NIL)) ; Дошли до списка свободных вершин
            (T (cons (print_lst (car ans)) (print_system (cdr ans))))))
(defun print_free(lst)(cond
```

```
   ((null lst) '(Свободных переменных нет))
            (T (cons '(Свободные переменные) (cons lst NIL)))
))
(defun print_lst(lst)(cons (car lst) (cons '= (beg (print_l (cdr lst) NIL)))))
(defun print_l(lst res)(cond
   ((null (cdr lst)) (print_koef (car lst) res))
   (T (print_l (cdr lst) (print_top (caar lst) (cadar lst) res)))
))
(defun print_koef(koef res)(cond
   ((null koef) res)
   ((< 0 (car koef)) (cons '+ (cons (/ (car koef) (cadr koef)) res)))
   ((> 0 (car koef)) (cons '- (cons (/ (* -1 (car koef)) (cadr koef)) res)))
   (T res) ; нулевой коэффициент
))
(defun print_top(top koef res)(cond
   ((null koef) res)
   ((= 1 (/ (car koef) (cadr koef))) (cons '+ (cons top res)))
   ((= -1 (/ (car koef) (cadr koef))) (cons '- (cons top res)))
   ((< 0 (car koef)) (cons '+ (cons (/ (car koef) (cadr koef)) (cons '* (cons top res)))))
   ((> 0 (car koef)) (cons '- (cons (/ (* -1 (car koef)) (cadr koef)) (cons '* (cons top res)))))
   (T res) ; нулевой коэффициент
))
(defun beg(lst)(cond
   ((null lst) (cons 0 NIL)) ; решение вида x = 0
   ((eq '+ (car lst)) (cdr lst))
   (T lst)
))
```

Тесты

1) Нет решений

(main((X – Y = 0)(X + Y = 1)(X = 2 * Y)))-> "Система не имеет решений"
(main((X + Y = Y)(X = 1)(Y = 2)))-> "Система не имеет решений"
(main '((Y = 3 - 2 X)(2 = -9))) -> "Система не имеет решений"
(main '((Y + 1 = 3 - 2 X)(X = Y)(X + 2 Y = 1)))-> "Система не имеет решений"

2) Одно решение

(main((X – Y = 0)(X = Y)(Z = 1)))-> ((Z = 1) (X = 0) (Y = 0) (СВОБОДНЫХ ПЕРЕМЕННЫХ НЕТ))
(main((X + Z = 1)(Y = X)(X + Y + Z = 2)))-> ((Z = 0) (Y = 1) (X = 1) (СВОБОДНЫХ ПЕРЕМЕННЫХ НЕТ))
(main '((X + 4 Y = -1)(X + Y = 2)))-> ((Y = - 1) (X = 3) (СВОБОДНЫХ ПЕРЕМЕННЫХ НЕТ))
(main '((Y = 3 - 2 X)(6 X + Y = -9)))-> ((Y = 9) (X = - 3) (СВОБОДНЫХ ПЕРЕМЕННЫХ НЕТ))
(main '((X + Y = 2)(X + Y + Z = 2)(Z = 0)(Y - Z = 1)))-> ((Z = 0) (X = 1) (Y = 1) (СВОБОДНЫХ ПЕРЕМЕННЫХ НЕТ))

3) Есть свободные переменные

(main((X = Y)(0 = 0)))-> ((Y = X) ((СВОБОДНЫЕ ПЕРЕМЕННЫЕ) (X)))
(main '((X + Y + Z = 1)(X = Y + 2)))-> ((Y = - 1/2 - 1/2 * Z) (X = 3/2 - 1/2 * Z) ((СВОБОДНЫЕ ПЕРЕМЕННЫЕ) (Z)))
(main '((X + Y + Z = 2)(X + Y - Z = 0)))-> ((Z = 1) (X = 1 - Y) ((СВОБОДНЫЕ ПЕРЕМЕННЫЕ) (Y)))