Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

**ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ОБРАЗОВАТЕЛЬНО – НАУЧНЫЙ ПРОЕКТ “Apply”**

По дисциплине

«Алгебра и анализ»

Решение СЛАУ методом обратной матрицы

**Выполнил студент**

Группы ПИ21-4

Кондрашов Дмитрий Михайлович

**Преподаватель**

Камышова Галина Николаевна

**Москва 2021**

Оглавление

[Математическая задача 3](#_Toc89903292)

[Алгоритм решения 5](#_Toc89903293)

[Внешний вид программы 7](#_Toc89903294)

[Листинг программы 8](#_Toc89903295)

[Использованные инструменты 15](#_Toc89903296)

[Инструкция работы с приложением 16](#_Toc89903297)

[Тестовые задачи для запуска 17](#_Toc89903298)

# Математическая задача

Пусть задана система, состоящая из m уравнений с n неизвестными вида:



где - заданные числа,

а - неизвестные



Решением системы называется такая совокупность чисел, при подстановке которых каждое уравнение системы обращается в верное тождество.

**Решение СЛАУ в матричном виде:**

Коэффициенты при неизвестных в уравнениях системы образуют матрицу размера, которую обозначим



и назовем матрицей системы.

Вектор - столбец неизвестных системы:



Вектор - столбец свободных членов системы:

Рассмотрим произведение матриц:



Таким образом, систему можно записать в матричном виде:

Рассмотрим случай, когда m=n, то есть количество уравнений в системе равно количеству неизвестных. Матрица системы А – квадратная матрица порядка n.



Будем считать, что матрица А – невырожденная матрица, то есть



Пусть система записывается в матричном виде



У матрицы А существует единственная матрица . Умножим слева обе части уравнения на матрицу



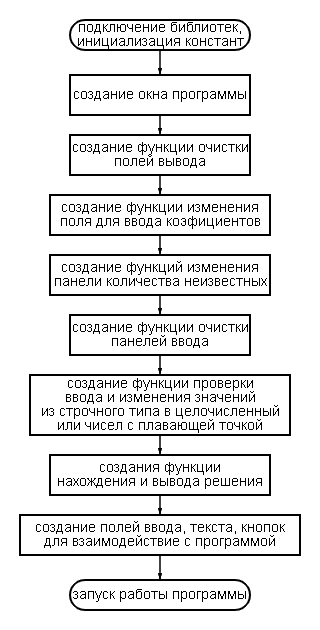
Таким образов, решением СЛАУ является вектор-столбец, который получается путём перемножения обратной матрицы коэффициентов при неизвестных на вектор-столбец свободных коэффициентов.

# Алгоритм решения

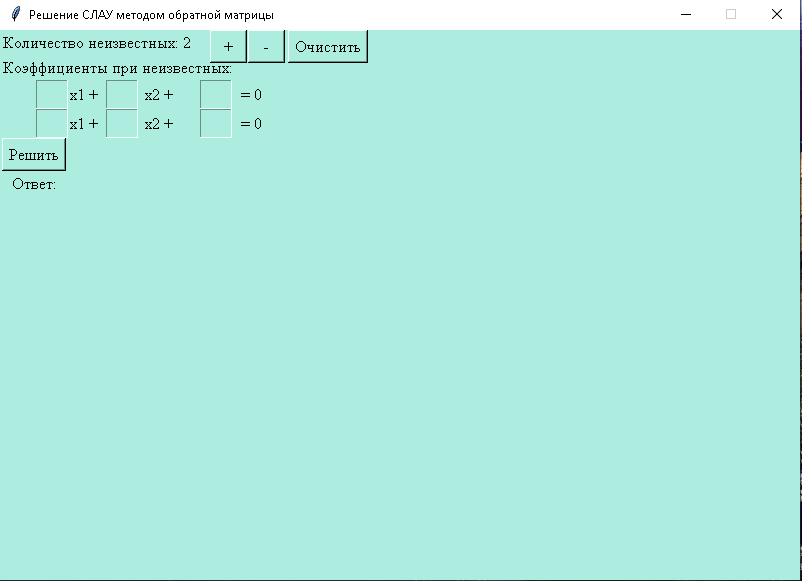
Алгоритм нахождения решения программным методам:

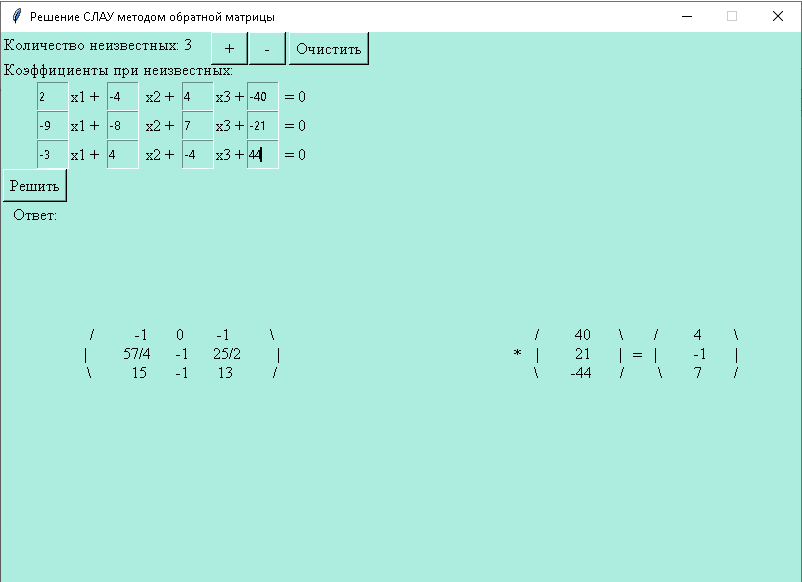
1. Пользователь вводит значения при неизвестных уравнений и значения свободных членов.
2. Программа проверяет корректность введённых данных
   1. были ли введены только целые или десятичные числа
   2. были ли заполнены все ячейки для ввода значений, чтобы получилась квадратная матрица
3. В случае некорректного ввода данных программа предупредит пользователя и будет ожидать ввода данных заново. Иначе программа инициализирует матрицу коэффициентов при неизвестных и вектор-столбец свободных коэффициентов.
4. Программа найдёт обратную матрицу к матрице коэффициентов при неизвестных в случае, если матрица не является вырожденной. Иначе она выведет ответ «коэффициентов при неизвестных» на интерфейс и будет ожидать дальнейших действий.
5. В случае, если существует решение СЛАУ, программа выведет обратную матрицу, перемноженную на определитель, вектор-столбец свободных коэффициентов и вектор-столбец решения и будет ожидать дальнейших действий (пользователь может очистить поле ответа и поля для ввода коэффициентов и ввести новые значения, либо закрыть программу).

Блок-схема алгоритма программы:



# Внешний вид программы





# Листинг программы

import numpy  
import tkinter  
import re  
import fractions  
count = 2  
norm\_digit = re.compile(r"-?\d+[,.]?\d\*")  
def remover():  
 matrix\_a['text'] = ''  
 entererror['text'] = ''  
 matrix\_a['text'] = ''  
 matrix\_b['text'] = ''  
 matrix\_result['text'] = ''  
 lable\_multiply['text'] = ''  
 lable\_equall['text'] = ''  
 matrix\_a.grid\_remove()  
 entererror.grid\_remove()  
 matrix\_a.grid\_remove()  
 matrix\_b.grid\_remove()  
 matrix\_result.grid\_remove()  
 lable\_multiply.grid\_remove()  
 lable\_equall.grid\_remove()  
def check\_count(x, how):  
 if x == 3 and how == '+':  
 x1enter3.grid(row=4, column=0, ipadx=5, ipady=5, stick='e')  
 x2enter3.grid(row=4, column=2, ipadx=5, ipady=5, stick='w')  
 x3enter1.grid(row=2, column=4, ipadx=5, ipady=5, stick='w')  
 x3enter2.grid(row=3, column=4, ipadx=5, ipady=5, stick='w')  
 x3enter3.grid(row=4, column=4, ipadx=5, ipady=5, stick='w')  
 x13.grid(row=4, column=1,stick='w')  
 x23.grid(row=4, column=3,stick='w')  
 x31.grid(row=2, column=5,stick='w')  
 x32.grid(row=3, column=5,stick='w')  
 x33.grid(row=4, column=5,stick='w')  
 eq3.grid(row=4, column=13,stick='w')  
 eq3enter.grid(row=4, column=12, ipadx=5, ipady=5, stick='w')  
 if x == 4 and how == '+':  
 x1enter4.grid(row=5, column=0, ipadx=5, ipady=5, stick='e')  
 x2enter4.grid(row=5, column=2, ipadx=5, ipady=5, stick='w')  
 x3enter4.grid(row=5, column=4, ipadx=5, ipady=5, stick='w')  
 x4enter1.grid(row=2, column=6, ipadx=5, ipady=5, stick='w')  
 x4enter2.grid(row=3, column=6, ipadx=5, ipady=5, stick='w')  
 x4enter3.grid(row=4, column=6, ipadx=5, ipady=5, stick='w')  
 x4enter4.grid(row=5, column=6, ipadx=5, ipady=5, stick='w')  
 x14.grid(row=5, column=1, stick='w')  
 x24.grid(row=5, column=3, stick='w')  
 x34.grid(row=5, column=5, stick='w')  
 x41.grid(row=2, column=7, stick='w')  
 x42.grid(row=3, column=7, stick='w')  
 x43.grid(row=4, column=7, stick='w')  
 x44.grid(row=5, column=7, stick='w')  
 eq4.grid(row=5, column=13, stick='w')  
 eq4enter.grid(row=5, column=12, ipadx=5, ipady=5, stick='w')  
 if x == 5 and how == '+':  
 x1enter5.grid(row=6, column=0, ipadx=5, ipady=5, stick='e')  
 x2enter5.grid(row=6, column=2, ipadx=5, ipady=5, stick='w')  
 x3enter5.grid(row=6, column=4, ipadx=5, ipady=5, stick='w')  
 x4enter5.grid(row=6, column=6, ipadx=5, ipady=5, stick='w')  
 x5enter1.grid(row=2, column=8, ipadx=5, ipady=5, stick='w')  
 x5enter2.grid(row=3, column=8, ipadx=5, ipady=5, stick='w')  
 x5enter3.grid(row=4, column=8, ipadx=5, ipady=5, stick='w')  
 x5enter4.grid(row=5, column=8, ipadx=5, ipady=5, stick='w')  
 x5enter5.grid(row=6, column=8, ipadx=5, ipady=5, stick='w')  
 x15.grid(row=6, column=1, stick='w')  
 x25.grid(row=6, column=3, stick='w')  
 x35.grid(row=6, column=5, stick='w')  
 x45.grid(row=6, column=7, stick='w')  
 x51.grid(row=2, column=9, stick='w')  
 x52.grid(row=3, column=9, stick='w')  
 x53.grid(row=4, column=9, stick='w')  
 x54.grid(row=5, column=9, stick='w')  
 x55.grid(row=6, column=9, stick='w')  
 eq5.grid(row=6, column=13, stick='w')  
 eq5enter.grid(row=6, column=12, ipadx=5, ipady=5, stick='w')  
 if x == 6 and how == '+':  
 x1enter6.grid(row=7, column=0, ipadx=5, ipady=5, stick='e')  
 x2enter6.grid(row=7, column=2, ipadx=5, ipady=5, stick='w')  
 x3enter6.grid(row=7, column=4, ipadx=5, ipady=5, stick='w')  
 x4enter6.grid(row=7, column=6, ipadx=5, ipady=5, stick='w')  
 x5enter6.grid(row=7, column=8, ipadx=5, ipady=5, stick='w')  
 x6enter1.grid(row=2, column=10, ipadx=5, ipady=5, stick='w')  
 x6enter2.grid(row=3, column=10, ipadx=5, ipady=5, stick='w')  
 x6enter3.grid(row=4, column=10, ipadx=5, ipady=5, stick='w')  
 x6enter4.grid(row=5, column=10, ipadx=5, ipady=5, stick='w')  
 x6enter5.grid(row=6, column=10, ipadx=5, ipady=5, stick='w')  
 x6enter6.grid(row=7, column=10, ipadx=5, ipady=5, stick='w')  
 x16.grid(row=7, column=1, ipadx=5, ipady=5, stick='w')  
 x26.grid(row=7, column=3, ipadx=5, ipady=5, stick='w')  
 x36.grid(row=7, column=5, ipadx=5, ipady=5, stick='w')  
 x46.grid(row=7, column=7, ipadx=5, ipady=5, stick='w')  
 x56.grid(row=7, column=9, ipadx=5, ipady=5, stick='w')  
 x61.grid(row=2, column=11, ipadx=5, ipady=5, stick='w')  
 x62.grid(row=3, column=11, ipadx=5, ipady=5, stick='w')  
 x63.grid(row=4, column=11, ipadx=5, ipady=5, stick='w')  
 x64.grid(row=5, column=11, ipadx=5, ipady=5, stick='w')  
 x65.grid(row=6, column=11, ipadx=5, ipady=5, stick='w')  
 x66.grid(row=7, column=11, ipadx=5, ipady=5, stick='w')  
 e6.grid(row=7, column=13, stick='w')  
 eq6enter.grid(row=7, column=12, ipadx=5, ipady=5, stick='w')  
  
 if x == 2 and how == '-':  
 for i in range (1, 3):  
 eval(f'x{i}enter3.grid\_remove()')  
 eval(f'x3enter{i}.grid\_remove()')  
 eval(f'x{i}3.grid\_remove()')  
 eval(f'x3{i}.grid\_remove()')  
 x3enter3.grid\_remove()  
 x33.grid\_remove()  
 eq3.grid\_remove()  
 eq3enter.grid\_remove()  
 if x == 3 and how == '-':  
 for i in range (1, 4):  
 eval(f'x{i}enter4.grid\_remove()')  
 eval(f'x4enter{i}.grid\_remove()')  
 eval(f'x{i}4.grid\_remove()')  
 eval(f'x4{i}.grid\_remove()')  
 x4enter4.grid\_remove()  
 x44.grid\_remove()  
 eq4.grid\_remove()  
 eq4enter.grid\_remove()  
 if x == 4 and how == '-':  
 for i in range (1, 5):  
 eval(f'x{i}enter5.grid\_remove()')  
 eval(f'x5enter{i}.grid\_remove()')  
 eval(f'x{i}5.grid\_remove()')  
 eval(f'x5{i}.grid\_remove()')  
 x5enter5.grid\_remove()  
 x55.grid\_remove()  
 eq5.grid\_remove()  
 eq5enter.grid\_remove()  
 if x == 5 and how == '-':  
 for i in range (1, 6):  
 eval(f'x{i}enter6.grid\_remove()')  
 eval(f'x6enter{i}.grid\_remove()')  
 eval(f'x{i}6.grid\_remove()')  
 eval(f'x6{i}.grid\_remove()')  
 x6enter6.grid\_remove()  
 x66.grid\_remove()  
 e6.grid\_remove()  
 eq6enter.grid\_remove()  
def cplus():  
 global count  
 if count < 6:  
 count += 1  
 lable3['text'] = f'{count}'  
 check\_count(count, '+')  
def cminus():  
 global count  
 if count > 2:  
 count -= 1  
 lable3['text'] = f'{count}'  
 check\_count(count, '-')  
def delete(): # полная очистка всех панелей  
 for i in range(1, 7):  
 eval(f'eq{i}enter.delete(0, tkinter.END)')  
 for j in range(1, 7):  
 eval(f'x{i}enter{j}.delete(0, tkinter.END)')  
 remover()  
def number\_maker(number):  
 def digits(stroka):  
 for letter in stroka:  
 if letter not in '1234567890-.,':  
 return False  
 return True  
 global norm\_digit  
 if not norm\_digit.search(number) is None and digits(number):  
 entered\_number = norm\_digit.search(number).group().replace(',', '.')  
 try:  
 entered\_number = int(entered\_number)  
 except ValueError:  
 entered\_number = float(entered\_number)  
 return entered\_number  
def solver():  
 global mat\_solve  
 remover()  
 big\_a = list()  
 big\_b = list()  
 for i in range(1, 7):  
 stroka\_list = []  
 big\_b.append([number\_maker(eval(f'eq{i}enter.get()'))])  
 for j in range(1, 7):  
 stroka\_list.append(number\_maker(eval(f'x{j}enter{i}.get()')))  
 if len(stroka\_list) != 0:  
 big\_a.append(stroka\_list)  
 big\_a = list(filter(lambda x: any([True for elem in x if not elem is None]), big\_a))  
 big\_b = list(filter(lambda x: any([True for elem in x if not elem is None]), big\_b))  
 for i in range(len(big\_a)):  
 big\_a[i] = list(filter(lambda x: True if not x is None else False, big\_a[i]))  
 q = len(big\_a) == len(big\_b) and len(big\_a) != 0 and len(big\_b) != 0  
 for i in range(len(big\_a)):  
 if len(big\_a[i]) != len(big\_a):  
 q = False  
 break  
 if not q:  
 remover()  
 entererror['text'] = 'Ошибка ввода'  
 entererror.place(x=80, y=290)  
 else:  
 entererror.grid\_remove()  
 big\_b = numpy.array(big\_b)  
 big\_a = numpy.array(big\_a)  
 try:  
 big\_a\_det = int(fractions.Fraction(numpy.linalg.det(big\_a)).limit\_denominator(1))  
 if big\_a\_det != 0:  
 try:  
 mat\_solve = numpy.linalg.solve(big\_a, big\_b)  
 except:  
 big\_a\_det = 'Error'  
 except:  
 big\_a\_det = 'Error'  
 if big\_a\_det == 0:  
 matrix\_a['text'] = 'Нет решений'  
 matrix\_a.place(x=80, y=290)  
 elif big\_a\_det == 'Error':  
 matrix\_a['text'] = 'Ошибка ввода'  
 matrix\_a.place(x=80, y=290) # панель ошибки ввода  
 else:  
 big\_a = numpy.linalg.inv(big\_a)  
 big\_a\_text, big\_b\_text, mat\_solve\_text = '', '', ''  
 for i in range(len(big\_a)): # печать обратной матрицы  
 if i == 0:  
 big\_a\_text += '/ '.ljust(8)  
 elif i == len(big\_a)-1:  
 big\_a\_text += '\\ '.ljust(8)  
 else:  
 big\_a\_text += '| '.ljust(8)  
 for j in range(len(big\_a)):  
 big\_a\_text += str(fractions.Fraction(big\_a[i][j]).limit\_denominator(100)).center(7)+' '  
 if i == 0:  
 big\_a\_text += ' \\\n'.rjust(8)  
 elif i == len(big\_a) - 1:  
 big\_a\_text += ' /\n'.rjust(8)  
 else:  
 big\_a\_text += ' |\n'.rjust(8)  
 matrix\_a['text'] = big\_a\_text  
 matrix\_a.place(x=80, y=290) # панель матрицы а  
 for i in range(len(big\_b)): # печать матрицы свободных членов  
 if i == 0:  
 big\_b\_text += '/'.ljust(8) + str(-big\_b[i][0]).center(5) + '\\\n'.rjust(8)  
 elif i == len(big\_b)-1:  
 big\_b\_text += '\\'.ljust(8) + str(-big\_b[i][0]).center(5) + '/\n'.rjust(8)  
 else:  
 big\_b\_text += '| '.ljust(8) + str(-big\_b[i][0]).center(5) + ' |\n'.rjust(8)  
 matrix\_b['text'] = big\_b\_text  
 lable\_multiply['text'] = '\*'  
 lable\_multiply.place(x=500, y=310)  
 matrix\_b.place(x=530, y=290) # панель матрицы свободных членов  
 for i in range(len(mat\_solve)): # печать матрицы ответа  
 if i == 0:  
 mat\_solve\_text += '/'.ljust(8) + str(-fractions.Fraction(mat\_solve[i][0]).limit\_denominator(100)).center(5) + '\\\n'.rjust(8)  
 elif i == len(big\_b)-1:  
 mat\_solve\_text += ' \\'.ljust(8) + str(-fractions.Fraction(mat\_solve[i][0]).limit\_denominator(100)).center(5) + '/\n'.rjust(8)  
 else:  
 mat\_solve\_text += '| '.ljust(8) + str(-fractions.Fraction(mat\_solve[i][0]).limit\_denominator(100)).center(5) + ' |\n'.rjust(8)  
 matrix\_result['text'] = mat\_solve\_text  
 lable\_equall['text'] = '='  
 lable\_equall.place(x=620, y=310)  
 matrix\_result.place(x=650, y=290) # панель матрицы результата  
  
win = tkinter.Tk()  
logo = tkinter.PhotoImage(file='math.png')  
win.iconphoto(False, logo)  
win.config(bg='#ADEDE0')  
win.title('Решение СЛАУ методом обратной матрицы')  
win.geometry("800x550+150+150")  
win.resizable(False, False)  
  
lable1 = tkinter.Label(win, text='Количество неизвестных:', background='#ADEDE0', font=('Times New Roman', 12))  
lable1.grid(row=0, column=0, stick='w', columnspan=4)  
lable2 = tkinter.Label(win, text='Коэффициенты при неизвестных:', background='#ADEDE0', font=('Times New Roman', 12))  
lable3 = tkinter.Label(win, text=f'{count}', background='#ADEDE0', font=('Times New Roman', 12))  
lable2.grid(row=1, column=0, stick='w', columnspan=13)  
lable3.place(x=180, y=0)  
button\_plus = tkinter.Button(win, text='+', background='#ADEDE0', font=('Times New Roman', 12), activebackground='#ADEDE0', command=cplus, width=3)  
button\_minus = tkinter.Button(win, text='-', background='#ADEDE0', font=('Times New Roman', 12), activebackground='#ADEDE0', command=cminus, width=3)  
button\_plus.place(x=210, y=0)  
button\_minus.place(x=248, y=0)  
solve = tkinter.Button(win, text='Решить', background='#ADEDE0', font=('Times New Roman', 12), activebackground='#ADEDE0', command=solver)  
solve.grid(row=8, column=0)  
answer = tkinter.Label(win, text='Ответ:', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=9, column=0)  
deleter = tkinter.Button(win, text='Очистить', background='#ADEDE0', font=('Times New Roman', 12), activebackground='#ADEDE0', command=delete)  
deleter.place(x=288, y=0)  
entererror = tkinter.Label(win, text='Ошибка ввода', background='#ADEDE0', font=('Times New Roman', 12))  
matrix\_a = tkinter.Label(win, background='#ADEDE0', font=('Times New Roman', 12))  
matrix\_b = tkinter.Label(win, background='#ADEDE0', font=('Times New Roman', 12))  
matrix\_result = tkinter.Label(win, background='#ADEDE0', font=('Times New Roman', 12))  
  
# 2 неизвестные  
x1enter1 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x1enter1.grid(row=2, column=0, ipadx=5, ipady=5, stick='e')  
x1enter2 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x1enter2.grid(row=3, column=0, ipadx=5, ipady=5, stick='e')  
x2enter1 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x2enter1.grid(row=2, column=2, ipadx=5, ipady=5, stick='w')  
x2enter2 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x2enter2.grid(row=3, column=2, ipadx=5, ipady=5, stick='w')  
x10 = tkinter.Label(win, width=3, text='x1 +', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=2, column=1,stick='w')  
x11 = tkinter.Label(win, width=3, text='x1 +', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=3, column=1,stick='w')  
x20 = tkinter.Label(win, width=3, text='x2 +', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=2, column=3,stick='w')  
x21 = tkinter.Label(win, width=3, text='x2 +', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=3, column=3,stick='w')  
eq1 = tkinter.Label(win, width=3, text='= 0', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=2, column=13,stick='w')  
eq2 = tkinter.Label(win, width=3, text='= 0', background='#ADEDE0', font=('Times New Roman', 12)).grid(row=3, column=13,stick='w')  
eq1enter = tkinter.Entry(win, width=3, background='#ADEDE0')  
eq1enter.grid(row=2, column=12, ipadx=5, ipady=5, stick='w')  
eq2enter = tkinter.Entry(win, width=3, background='#ADEDE0')  
eq2enter.grid(row=3, column=12, ipadx=5, ipady=5, stick='w')  
  
# 3 неизвестные  
x1enter3 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x2enter3 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x3enter1 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x3enter2 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x3enter3 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x13 = tkinter.Label(win, width=3, text='x1 +', background='#ADEDE0', font=('Times New Roman', 12))  
x23 = tkinter.Label(win, width=3, text='x2 +', background='#ADEDE0', font=('Times New Roman', 12))  
x31 = tkinter.Label(win, width=3, text='x3 +', background='#ADEDE0', font=('Times New Roman', 12))  
x32 = tkinter.Label(win, width=3, text='x3 +', background='#ADEDE0', font=('Times New Roman', 12))  
x33 = tkinter.Label(win, width=3, text='x3 +', background='#ADEDE0', font=('Times New Roman', 12))  
eq3 = tkinter.Label(win, width=3, text='= 0', background='#ADEDE0', font=('Times New Roman', 12))  
eq3enter = tkinter.Entry(win, width=3, background='#ADEDE0')  
  
#4 неизвестные  
x1enter4 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x2enter4 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x3enter4 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x4enter1 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x4enter2 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x4enter3 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x4enter4 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x14 = tkinter.Label(win, width=3, text='x1 +', background='#ADEDE0', font=('Times New Roman', 12))  
x24 = tkinter.Label(win, width=3, text='x2 +', background='#ADEDE0', font=('Times New Roman', 12))  
x34 = tkinter.Label(win, width=3, text='x3 +', background='#ADEDE0', font=('Times New Roman', 12))  
x41 = tkinter.Label(win, width=3, text='x4 +', background='#ADEDE0', font=('Times New Roman', 12))  
x42 = tkinter.Label(win, width=3, text='x4 +', background='#ADEDE0', font=('Times New Roman', 12))  
x43 = tkinter.Label(win, width=3, text='x4 +', background='#ADEDE0', font=('Times New Roman', 12))  
x44 = tkinter.Label(win, width=3, text='x4 +', background='#ADEDE0', font=('Times New Roman', 12))  
eq4 = tkinter.Label(win, width=3, text='= 0', background='#ADEDE0', font=('Times New Roman', 12))  
eq4enter = tkinter.Entry(win, width=3, background='#ADEDE0')  
  
# 5 неизвестных  
x1enter5 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x2enter5 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x3enter5 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x4enter5 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x5enter1 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x5enter2 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x5enter3 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x5enter4 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x5enter5 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x15 = tkinter.Label(win, width=3, text='x1 +', background='#ADEDE0', font=('Times New Roman', 12))  
x25 = tkinter.Label(win, width=3, text='x2 +', background='#ADEDE0', font=('Times New Roman', 12))  
x35 = tkinter.Label(win, width=3, text='x3 +', background='#ADEDE0', font=('Times New Roman', 12))  
x45 = tkinter.Label(win, width=3, text='x4 +', background='#ADEDE0', font=('Times New Roman', 12))  
x51 = tkinter.Label(win, width=3, text='x5 +', background='#ADEDE0', font=('Times New Roman', 12))  
x52 = tkinter.Label(win, width=3, text='x5 +', background='#ADEDE0', font=('Times New Roman', 12))  
x53 = tkinter.Label(win, width=3, text='x5 +', background='#ADEDE0', font=('Times New Roman', 12))  
x54 = tkinter.Label(win, width=3, text='x5 +', background='#ADEDE0', font=('Times New Roman', 12))  
x55 = tkinter.Label(win, width=3, text='x5 +', background='#ADEDE0', font=('Times New Roman', 12))  
eq5 = tkinter.Label(win, width=3, text='= 0', background='#ADEDE0', font=('Times New Roman', 12))  
eq5enter = tkinter.Entry(win, width=3, background='#ADEDE0')  
  
# 6 неизвестных  
x1enter6 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x2enter6 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x3enter6 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x4enter6 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x5enter6 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x6enter1 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x6enter2 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x6enter3 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x6enter4 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x6enter5 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x6enter6 = tkinter.Entry(win, width=3, background='#ADEDE0')  
x16 = tkinter.Label(win, width=3, text='x1 +', background='#ADEDE0', font=('Times New Roman', 12))  
x26 = tkinter.Label(win, width=3, text='x2 +', background='#ADEDE0', font=('Times New Roman', 12))  
x36 = tkinter.Label(win, width=3, text='x3 +', background='#ADEDE0', font=('Times New Roman', 12))  
x46 = tkinter.Label(win, width=3, text='x4 +', background='#ADEDE0', font=('Times New Roman', 12))  
x56 = tkinter.Label(win, width=3, text='x5 +', background='#ADEDE0', font=('Times New Roman', 12))  
x61 = tkinter.Label(win, width=3, text='x6 +', background='#ADEDE0', font=('Times New Roman', 12))  
x62 = tkinter.Label(win, width=3, text='x6 +', background='#ADEDE0', font=('Times New Roman', 12))  
x63 = tkinter.Label(win, width=3, text='x6 +', background='#ADEDE0', font=('Times New Roman', 12))  
x64 = tkinter.Label(win, width=3, text='x6 +', background='#ADEDE0', font=('Times New Roman', 12))  
x65 = tkinter.Label(win, width=3, text='x6 +', background='#ADEDE0', font=('Times New Roman', 12))  
x66 = tkinter.Label(win, width=3, text='x6 +', background='#ADEDE0', font=('Times New Roman', 12))  
e6 = tkinter.Label(win, width=3, text='= 0', background='#ADEDE0', font=('Times New Roman', 12))  
eq6enter = tkinter.Entry(win, width=3, background='#ADEDE0')  
  
lable\_multiply = tkinter.Label(win, width=3, text='\*', background='#ADEDE0', font=('Times New Roman', 12))  
lable\_equall = tkinter.Label(win, width=3, text='=', background='#ADEDE0', font=('Times New Roman', 12))  
win.mainloop()

# Использованные инструменты

В программе были использованы стандартные инструменты языка Python, такие как функции, методы различных типов данных, циклы for и while, проверки if, elif, else.

Также были подключены три встроенные в компилятор библиотеки и одна дополнительная.

1. Numpy – библиотека, необходимая для работы с матрицами чисел. Использовались функции инициализации матриц, нахождения определителя, нахождения обратной матрицы, перемножения матриц и решения СЛАУ при наличии матриц коэффициентов при неизвестных и матрицы свободных членов. **Данную библиотеку необходимо установить командой pip install numpy, введённой в терминал компилятора (в случае её отсутствия) для корректной работы программы.**
2. Tkinter – библиотека для создания графического пользовательского интерфейса. Нужна для удобства взаимодействия с программой, путём нажатия «кнопок» в окне приложения, ввода данных с клавиатуры в специальные ячейки. Динамически изменяет внешний вид программы во время взаимодействия.
3. Re – библиотека для проверки строки на соответствие заданному паттерну. С её помощью проверяются введённые с клавиатуры значения. С её помощью предотвращаются ошибки работы программы.
4. Fractions – библиотека обыкновенных дробей. Преобразует десятичные числа в простые дроби для удобства вывода ответа (иногда при решения получаются иррациональные числа, которые лучше вывести на печать в качестве обыкновенных дробей).

# Инструкция работы с приложением

При запуске программы появляется окно формата GUI. На нём имеется несколько ячеек для ввода данных и 4 кнопки. По стандарту на экране всего 6 ячеек для ввода данных, их количество можно изменять в зависимости от количества неизвестных СЛАУ(оно меняется от 2 до 6).

При нажатии на кнопку «+» количество ячеек ввода увеличится. При достижении числа 6 вверху экрана нажатие перестанет давать какой-либо эффект.

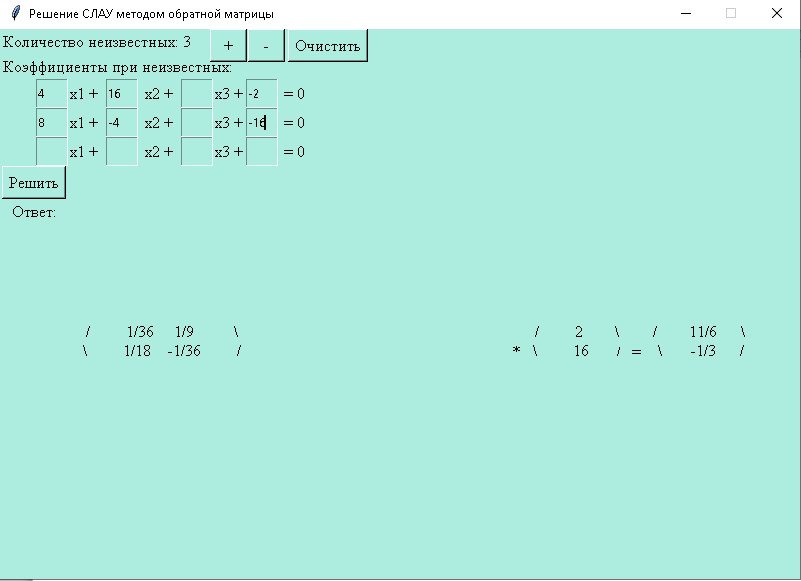
При нажатии на кнопку «-»количество ячеек ввода уменьшится. При достижении числа 2 вверху экрана нажатие перестанет давать какой-либо эффект.

При нажатии на кнопку «Очистить» все поля ввода будут очищены (все введённые значения будут стёрты), также очистятся поля вывода, если они присутствуют на экране.

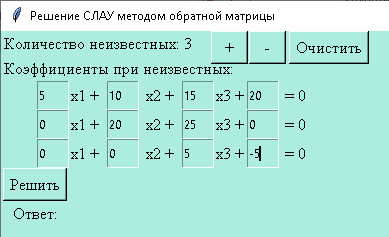
При нажатии на кнопку «Решить» внизу экрана отобразится поле вывода. При нажатии на кнопку повторно без изменения введённых данных не будет никакого эффекта.

Для ввода данных в ячейки ввода нужно навестись на ячейку и нажать правой клавишей мыши на неё, после чего вводить значение с клавиатуры.

При увеличении или уменьшение количества ячеек ввода введённые значения (если таковые имеются) могут быть скрыты, вследствие чего, при обновлении данных можно не заметить, что заполнена скрытая ячейка и программа может инициализировать матрицу с лишним введённым числом и будет выводить сообщение о некорректном вводе. Поэтому после решения одной СЛАУ советуется очищать все поля с помощью кнопки. Также имеется возможность вводить коэффициенты матрицы меньшего размера, чем на экране (например, решать СЛАУ с двумя неизвестными, когда на экране больше ячеек, чем нужно), в таком случае следует оставлять поля ввода пустыми для отсутствующих коэффициентов.

Например, вот так: 

В уравнениях с нулевыми коэффициентами необходимо вводить 0 у неизвестных.

Например, вот так: 

# Тестовые задачи для запуска

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.