Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

Московский техникум космического приборостроения

(МТКП МГТУ имени Н.Э. Баумана)

СПЕЦИАЛЬНОСТЬ 09.02.03 Программирование в компьютерных системах

П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовому проекту на тему:

РАЗРАБОТКА ПРОГРАММЫ РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДАМИ ВРАЩЕНИЯ, РАЗЛОЖЕНИЯ НА ПРОИЗВЕДЕНИЕ ДВУХ ТРЕУГОЛЬНЫХ МАТРИЦ

Студент _____ А.В. Соловых
подпись

Руководитель курсового проекта _____ П.В. Русанов
подпись

Москва 2016

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
Московский техникум космического приборостроения
(МТКП МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Председатель ПЦК специальности 230115

Жилкина Н.А.
«_____»_____2015 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине МДК.03.01 Технология разработки программного обеспечения

Студент Соловых А.В ТМП-62
(фамилия, инициалы, индекс группы)

Руководитель Русанов П.В.
(фамилия, инициалы)

График выполнения работы: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 15 нед.

1. Тема курсового проекта

Разработка программы решения системы линейных уравнений методами вращения, разложения на произведение двух треугольных матриц

2. Техническое задание

2.1 Написать программу для решения СЛАУ методом Якоби
2.2 Написать программу для обращения матрицы методом LU-разложения

3. Оформление курсового проекта

3.1. Пояснительная записка на_____ листах формата А4.
3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.) – схемы алгоритма программы

Дата выдачи задания «__29__»_____января_____2016 г.

Руководитель курсового проекта _____ Русанов П.В.

Содержание

ВВЕДЕНИЕ

1 Матрицы

Матрицей называется прямоугольная таблица чисел.

$$A = \begin{pmatrix} 1 & 2 & 7 \\ 0 & 1 & -7 \\ 8 & 9 & 0 \end{pmatrix}$$

Матрицы обозначаются заглавными полужирными буквами (A), а их элементы — соответствующими строчными буквами с индексами, т.е. aij. Первый индекс нумерует строки, а второй — столбцы. Принято обозначать максимальное значение индекса той же буквой, что и сам индекс, но заглавной. Поэтому матрицу A можно также записать как {aij, i = 1,..., I; j = 1,..., J}.

Пара чисел I и J называется размерностью матрицы и обознается как I×J.

2 Единичная матрица

Единичная матрица — такая квадратная матрица, элементы главной диагонали которой равны единице, а остальные равны нулю. Обычно единичную матрицу обозначают буквой E:

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Единичная матрица играет такую же роль, которую играет число 1 при перемножении вещественных чисел.

3 Обратная матрица

Обратная матрица — такая матрица A^{-1}, при умножении на которую исходная матрица A даёт в результате единичную матрицу E:

$$A \times A^{-1} = A^{-1} \times A = E$$

Квадратная матрица обратима тогда и только тогда, когда она невырожденная, то есть её определитель не равен нулю. Для неквадратных матриц и вырожденных матриц обратных матриц не существует. В данной работе представлен метод обращения матрицы методом окаймления.

Необходим переход к следующему разделу… «В данной курсовой работе…»

1 Постановка задачи

1.1 Написать программу для обращения матрицы методом Якоби

1.2 Написать программу для обращения матрицы методом LU-разложения

1.1 Обращение матрицы методом Якоби

Исходную систему Ax=b преобразуем к виду(1.1):

$$x_i = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j - \sum_{j=i+1}^{m} \frac{a_{ij}}{a_{ii}} x_j + \frac{b_i}{a_{ii}}$$

где i=1,2,...,m;$a_{ii} \neq 0$.

Первая сумма равна нулю, если верхний предел суммирования меньше нижнего.

Так (1.1) при i=1 имеет вид

$$x_i = -\sum_{j=2}^{m} \frac{a_{1j}}{a_{11}} x_j + \frac{b_1}{a_{11}}.$$

По методу Якоби x_i^{n+1} (n+1 приближение x_i) ищем по формуле(1.2):

$$x_i = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^n - \sum_{j=i+1}^{m} \frac{a_{ij}}{a_{ii}} x_j^n + \frac{b_i}{a_{ii}}$$

где n – номер итерации (0,1,…); i= $\overline{1,m}$.

Итерационный процесс (1.2) начинается с начальных значений x_i^0, которые в общем случае задаются произвольно, но предпочтительнее за x_i^0 взять свободные члены исходной системы.

Условие окончания счета:

$$max_i \left| x_i^{n+1} - x_i^m \right| < \varepsilon, \text{ где i} = \overline{1,m}.$$

1.2 Обращение матрицы методом LU-разложения

Будет использоваться метод Doolittle's LUP для разложения матрицы A в PA=LU, где L это нижняя треугольная матрица, U это верхняя треугольная матрица и P – матрица перестановок. P необходима, чтобы решить некоторые проблемы сингулярности.

Формула для просчета верхней треугольной матрицы(U):

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} u_{kj} l_{ik}$$

Формула для просчета L почти такая же, за исключением того что нужно разделить каждый член по соответствующему диагональному элементу U. Для того, чтобы убедиться, что алгоритм численно устойчив при $u_{jj} < 0$, матрица P используется для изменения порядка A так, чтобы наибольший элемент каждого столбца A сдвигался к диагонали A. Сама формула:

$$L_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} u_{kj} \cdot L_{ik} \right)$$

Commented [U1]: без личных местоимений

2 Структура и описание программы

Структура программы приведена на рисунке 2.1.

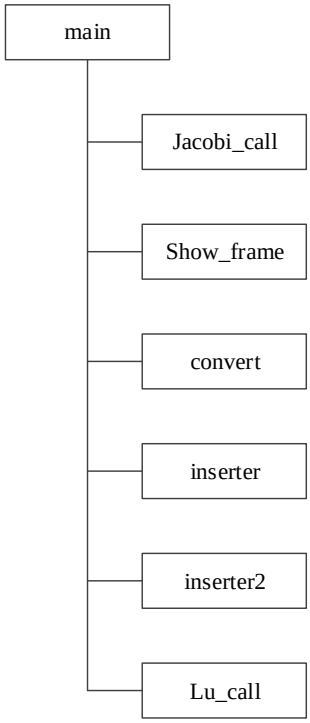


Рисунок 2.1 — Структура программы

Описание и назначение используемых подпрограмм приведено в таблице 2.1.

Таблица 2.1 – Используемые подпрограммы

Подпрограмма	Назначение
jacobi_call()	Построение интерфейса для ввода данных
inserter(value)	Функция вывода переданных значений
convert()	Получение значений и передача их функции
lu_call()	Построение интерфейса для ввода данных
show_frame(self, page_name)	Функция вызова страницу по переданному названию
inserter2(value, value2)	Функция вывода переданных значений

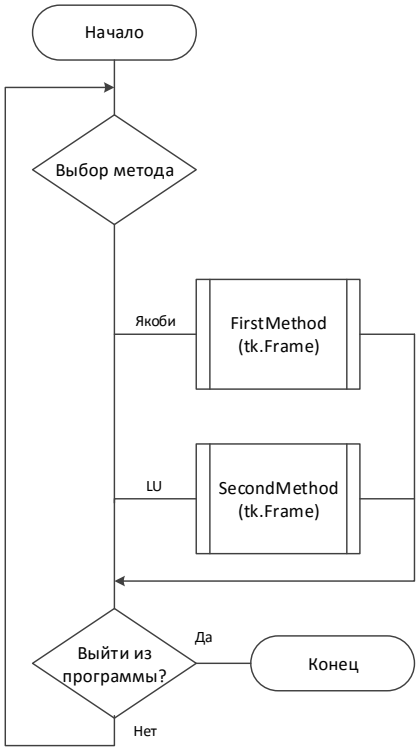
Описание и назначение используемых переменных приведено в таблице 2.2.

Таблица 2.2 – Используемые переменные

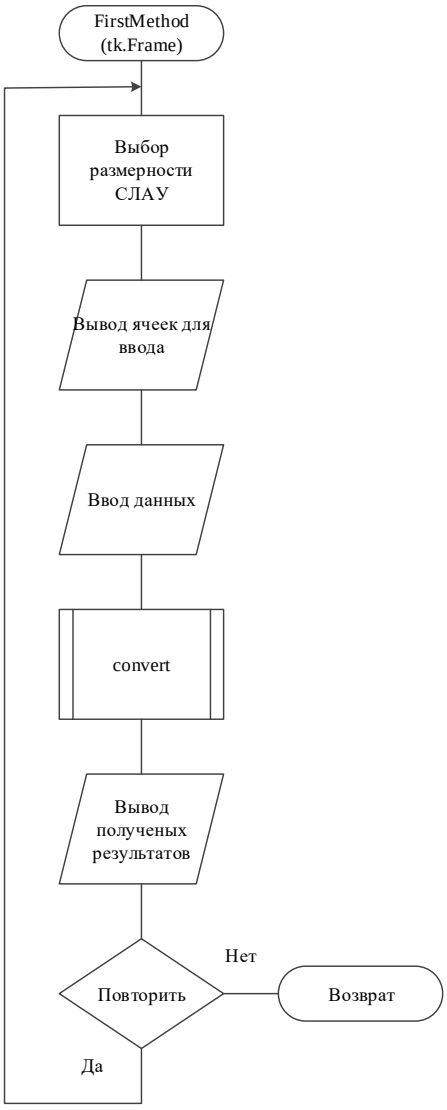
Переменная	Назначение
frame	Страница
Label…	Метка с текстом
Button…	Кнопки
n	Размерность матрицы/уравнений
Arr_txt	Ячейка для заполнения данными
arr_txt1	Вспомогательная ячейка для заполнения данными
arr_ntr	Массив значений
arr_ntr1	Вспомогательный массив значений
dop_infa	Метка с дополнительной информацией
helping_label…	Вспомогательная метка с текстом (подсказками)

3 Схемы алгоритма программы

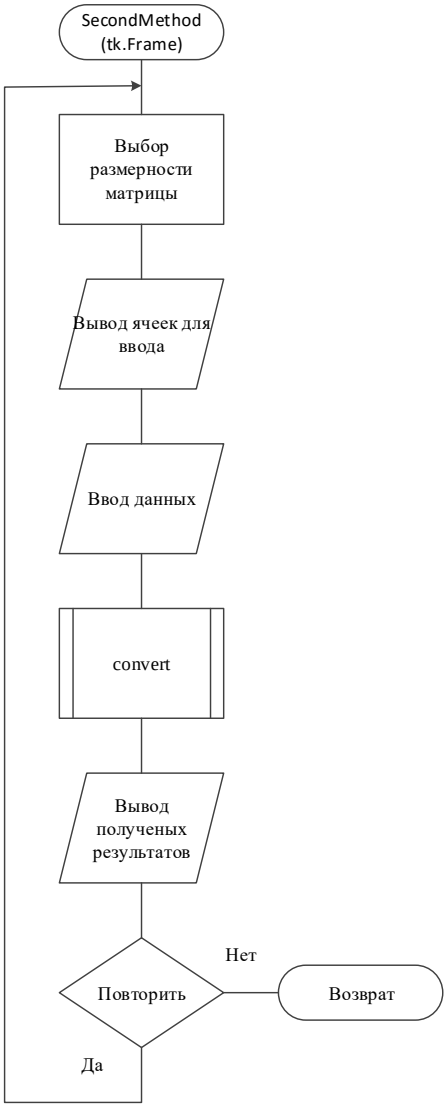
3.1 Схема алгоритма основной программы



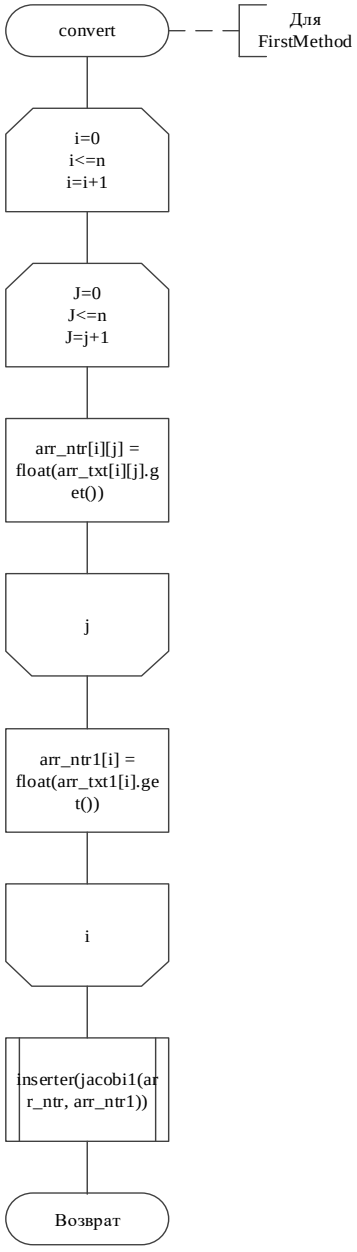
3.2 Схема алгоритма функции отображения 2-й страницы

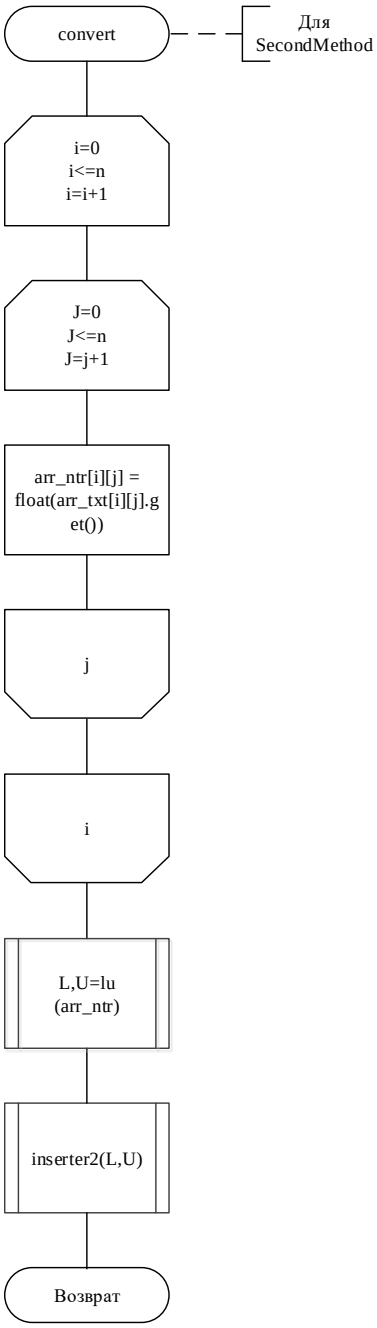


3.3 Схема алгоритма функции отображения 3-ей страницы

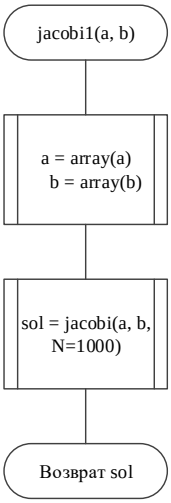


3.4 Схема алгоритма функции convert

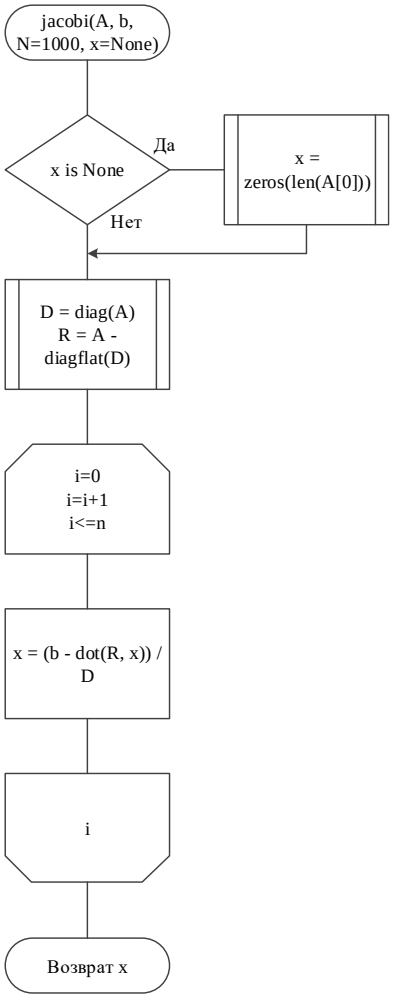



convert
Для SecondMethod
i=0
i<=n
i=i+1
J=0
J<=n
J=j+1
arr_ntr[i][j] = float(arr_txt[i][j].get())
j
i
L,U=lu (arr_ntr)
inserter2(L,U)
Возврат

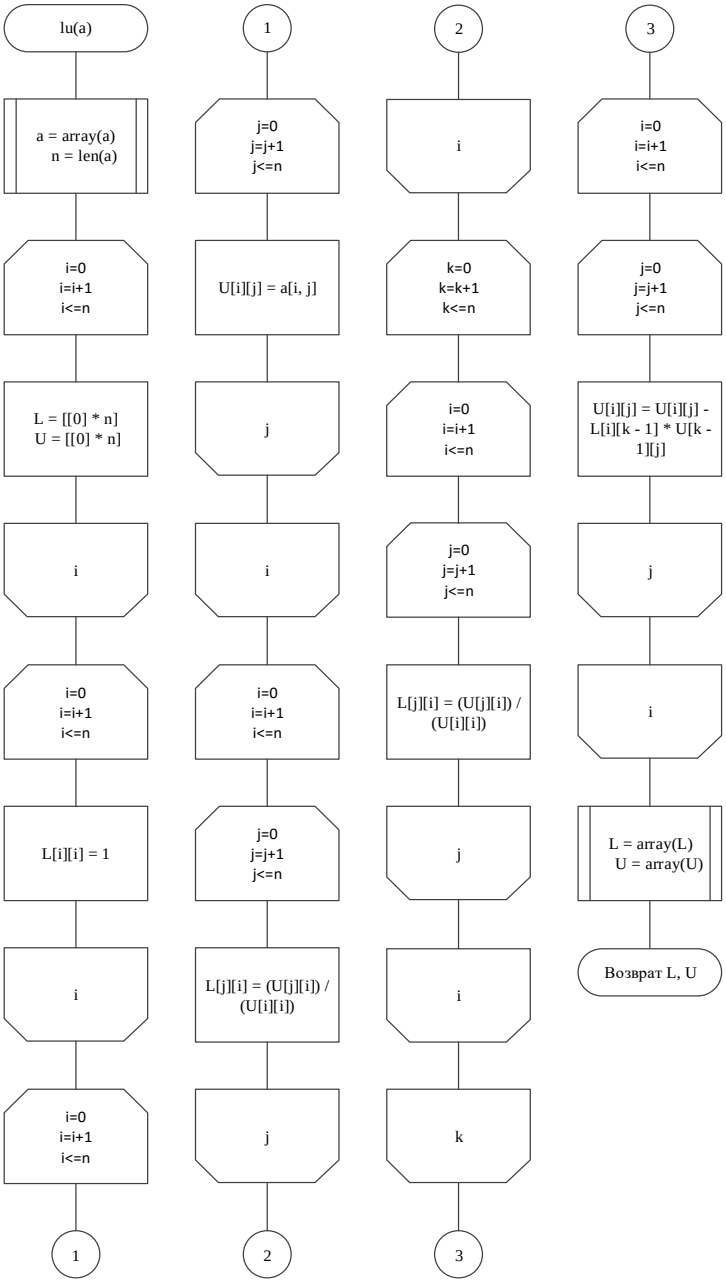
3.5 Схема алгоритма функции jacobi1



3.6 Схема алгоритма функции jacobi



3.7 Схема алгоритма функции LU



4 Отладка программы

Отладка – процесс локализации и исправления ошибок в программном коде. Этот процесс занимает значительную часть разработки, нередко - большую, по сравнению с составлением программы. Практически любая программа перед началом отладки содержит ошибки.

Синтаксические ошибки – ошибки, фиксируемые компилятором (транслятором, интерпретатором) при выполнении синтаксического и, частично, семантического анализа программы. Заключаются в несовпадении исходного кода с синтаксисом выбранного языка программирования.

Ошибки компоновки – ошибки, обнаруженные компоновщиком (редактором связей) при объединении модулей программы. Заключаются в несовпадении каких-либо связей при подключении модулей и подпрограмм или при обращении к ним.

Ошибки выполнения – происходят во время вычислений и чаще всего приводят к завершению работы программы. Самый распространенный пример такой ошибки - деление на ноль.

Ошибки логики – не позволяют программе выполнять предполагаемые действия. Код может компилироваться и выполняться без ошибок, но результат операции может оказаться неожиданным и неверным.

Во время работы были выявлены ошибки:

1) Логическая ошибка:

не было проверки при вводе размерности матрицы.

2) Ошибка выполнения:

ошибка обращения к функции (ошибка в написании имени функции).

3) Синтаксическая ошибка:

проблемы с отступами в разных IDE.

Все ошибки были исправлены. Для того чтобы убедиться в правильности работы программы, необходимо провести ее тестирование.

5 Оптимизация программы

В процессе разработки программы возникла необходимость в оптимизации алгоритма программы. Оптимизация проводилась по времени выполнения программы и по используемой памяти в процессе выполнения.

В процессе оптимизации по времени были выполнены следующие шаги:

1) реорганизация циклов таким образом, чтобы снизить количество инициализаций;

2) внутренняя оптимизация циклов, чтобы все повторяющиеся вычисления производились снаружи цикла;

3) повторяющиеся блоки операторов были оформлены в виде подпрограмм;

Таким образом, удалось снизить время выполнения программы, а также объем выделяемой памяти.

Для сокращения времени выполнения программы условный оператор в программе был вынесен за цикл, что сократило количество инициализаций и завершений цикла.

6 Тестирование программы

Тестирование — набор процедур и действий, предназначенных для демонстрации правильности работы программы в заданных режимах и внешних условиях. Цель тестирования - выявить наличие ошибок или продемонстрировать отсутствие ошибок и правильность работы программы.

Для того чтобы протестировать составленную программу, есть три способа тестирования, которые и были применены к данной программе:

1) тестирование в нормальных условиях;
2) тестирование в экстремальных условиях;
3) тестирование в исключительных условиях;

6.1 Тестирование в нормальных условиях

Для проведения тестирования в нормальных условиях, воспользуемся входными данными, которые лежат внутри диапазона допустимых значений. В данном случае диапазон допустимых значений для размерности матрицы лежит в границах для метода Якоби – от 2x2 до 4x4, и для LU разложения – от 2x2 до 5x5.

Для проверки была выбрана размерность 3x3 и 4x4 соответственно.

Ввод и вывод для метода Якоби представлен на рисунке 6.1.

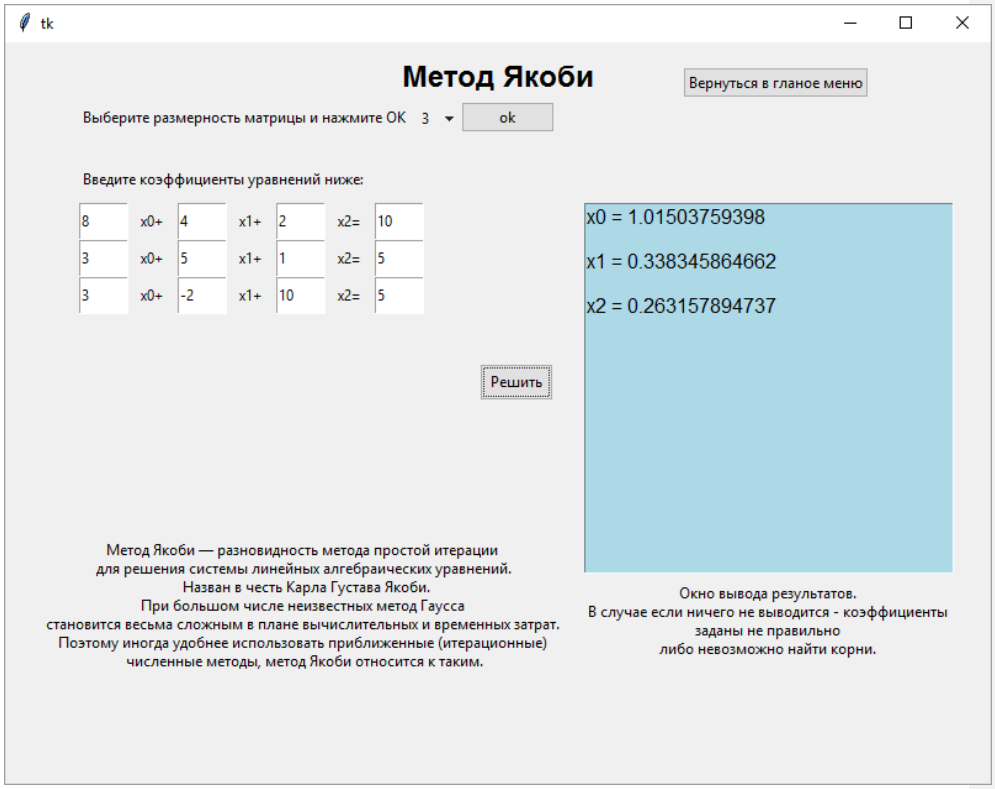


Рисунок 6.1 – Интерфейс 1-го метода

Ввод и вывод для метода Якоби представлен на рисунке 6.2.

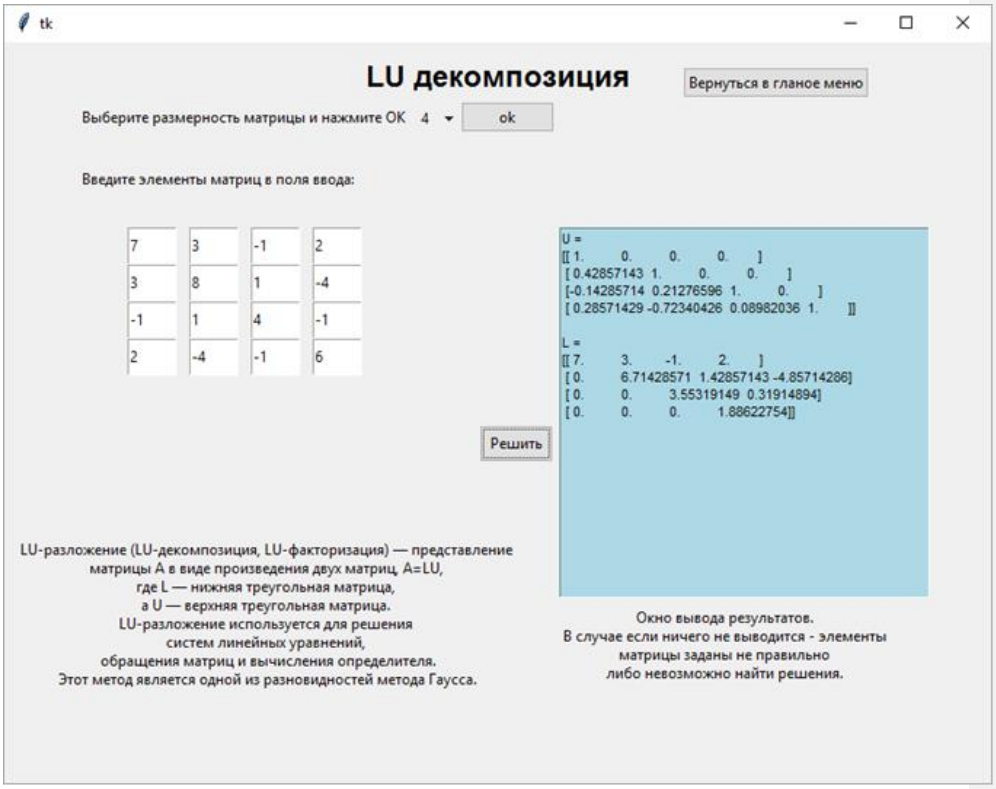


Рисунок 6.2 – Интерфейс 2-го метода

Произведем ручной просчет для проверки работы программы.

Для метода Якоби в таблице 6.1, а для метода LU разложения в таблице 6.2.

Таблица 6.1 – Метод Якоби

x0	x1	x2	=
8	4	2	10
3	5	1	5
3	-2	10	5
	x0	=	1,015
	x1	=	0,338
	x2	=	0,263

Таблица 6.2 – Метод LU разложения

7	3	-1	2	
3	8	1	-4	
-1	1	4	-1	
2	-4	-1	6	
U =	7	3	-1	2
	0	6,714	1,4285	-4,857
	0	0	3,553	0,319
	0	0	0	1,886
L =	1	0	0	0
	0,4285	1	0	0
	-0,143	0,213	1	0
	0,286	-0,723	0,09	1

Результат работы программы совпал с результатом ручного просчета, следовательно, программа в нормальных условиях работает правильно.

6.2 Тестирование в экстремальных условиях

На рисунке 6.3 представлены экстремальные значения элементов матрицы.

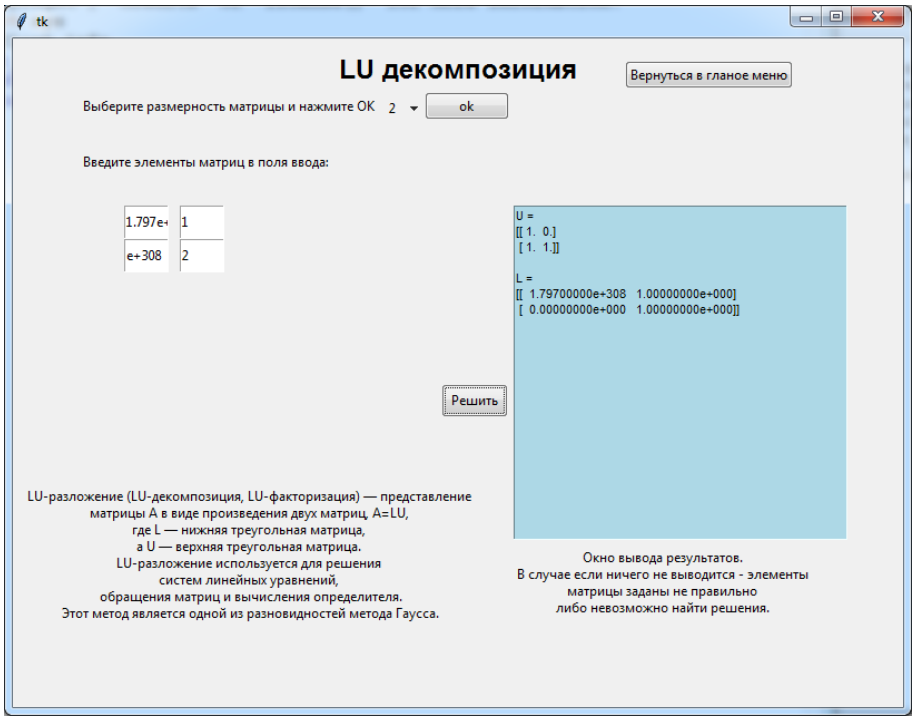


Рисунок 6.3 – Ввод экстремальных значений.

При вводе экстремальных значений программа продолжает работает и не ломается, и пользователь может продолжить работу.

Соответсвенно программа в экстремальных условиях работает правильно.

6.3 Тестирование в исключительных условиях

Ввод некорректных данных представлен на рисунке 6.4.

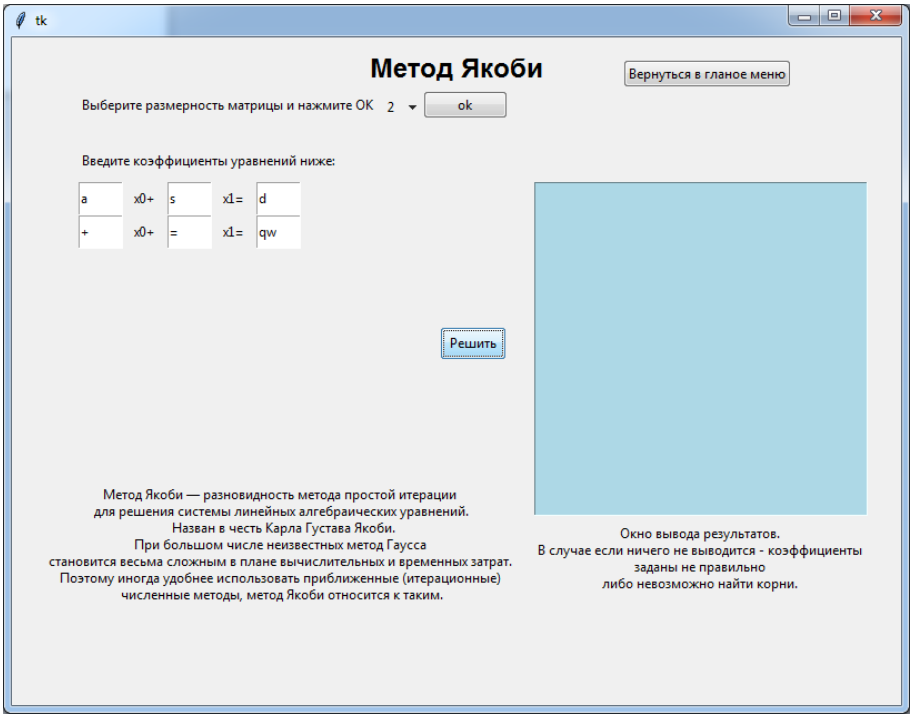


Рисунок 6.4 – Ввод некорректных значений.

При вводе неправильных данных программа не ломается и пользователь может продолжить работу.

Программа в исключительных условиях работает правильно.

Так как программа прошла все этапы тестирования, следовательно, программа работает нормально во всех условиях.

Листинг программы приведен в приложении А, а результаты выполнения в приложении Б.

ЗАКЛЮЧЕНИЕ

По итогам выполнения курсового проекта была получена программа с возможностью решать СЛАУ методом Якоби и обращать матрицы с помощью LU разложения.

В процессе разработки были выявлены некоторые недостатки разработанной программы. К ним можно отнести ограничение размерности вводимой матрицы, обусловленное максимальным размером массива.

К достоинствам программы можно отнести оптимизацию кода, наличие проверок, вводимых данных, скорость выполнения и универсальность программы.

Проведенное тестирование позволило исправить все ошибки и приобрести опыт в отладке программ.

Также были получены навыки составления документации.

Список использованной литературы

1 Данилина Н.И., Дубровская Н.С.. Численные методы. Высшая школа, 2012

2 Мартин Р. Идеальный программист. Издательство «Питер» , 2012

3 Уоррен Г. Алгоритмические трюки для программистов. Издательство «Вильяме», 2014

4 И. Г. Семакин, А. П. Шестаков Основы программирования 2006

5 Шелест В. Программирование. Издательство «БХВ-Петербург», 2009

6 Уэзерелл Ч. Этюды программистов. Издательство «Мир», 2008

7 Макконнелл С. Совершенный код. Издательство «Русская редакция», 2010

8 Очков В.Ф., Пухначев Ю. В. 128 советов начинающему программисту 2011

9 Дунаев С. Технологии Интернет-программирования. Издательство «БХВ-Петербург», 2006

10 В.Н. Тарасов, Н.Ф. Бахарева Численные методы, теории, алгоритмы, программы. Издательство «Самара», 2008.

11 Ram K. C Development with Eclipse. Издательство «Packt Publishing», 2015

12 Данилина Н.И., Дубровская Н.С. Численные методы. Издательство «Высшая школа», 1976

13 Семакин И. Г., Шестаков А. П. Основы программирования. Издательство «Высшая школа, НМЦ СПО, Мастерство», 2001

14 Лутц М. Python Полное руководство том 1, 4-е издание. Издательство «Вильямс», 2010

15 Лутц М. Python Полное руководство том 2, 4-е издание. Издательство «Вильямс», 2010

Commented [U2]: Паскаль?

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программы

```
# -*- coding: utf-8 -*-
"""
tkinter - библиотека с функциями для построения простого интерфейса.
       ttk - расширение библиотеки tkinter, улучшающая графику;

Курсовой проект
Предмет "МДК.03.01 Технология разработки программного обеспечения"
Тема "Разработка программы решения СЛАУ методами Якоби и методом LU
разложения"
Язык: Python 3.4
Разработал Соловых Андрей Викторович. Группа ТМП62.
Версия 4.1, 03.05.2016.

Задача:
Написать программу для решения СЛАУ методами Якоби и методом LU
разложения
"""


import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from metod_vrasheniya20 import jacobi1
from LU_decomp import lu


TITLE_FONT = ("Helvetica", 18, "bold")
DOP_FONT = ("Consolas",15,"bold")
DOP_FONT2 = ("Arial",13,"bold")


class SampleApp(tk.Tk):


    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        # контейнер в котором мы храним наши фреймы
        # по порядку
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (MainMenu, FirstMethod, SecondMethod):
            page_name = F.__name__
            frame = F(container, self)
            self.frames[page_name] = frame
```

```
            # put all of the pages in the same location;
            # the one on the top of the stacking order
            # will be the one that is visible.
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame("MainMenu")

    def show_frame(self, page_name):
        """ Вызывает страницу по переданому названию """
        frame = self.frames[page_name]
        frame.tkraise()


class MainMenu(tk.Frame):
    """ Страница основного меню """


    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        label = tk.Label(self, text="Меню", font=TITLE_FONT)
        label.pack(side="top", fill="x", pady=10)
        str_inf = "Курсовой проект\n Разработчик : Соловых А.В \n
Группа : ТМП62\n Задание: Разработка программы решения системы
линейных уравнений:\n -метод вращения(метод якоби) \n -метод LU
разложения матриц"
        label2 = tk.Label(self, text=str_inf,font=DOP_FONT)
        label2.place(x=40, y=340)
        label11 = tk.Label(self, text="Выберите нужный
метод:",font=DOP_FONT2)
        label11.pack()
        button1 = ttk.Button(self, text="Метод Якоби(вращения
матриц)",
                             command=lambda:
controller.show_frame("FirstMethod"))
        button2 = ttk.Button(self, text="Метод LU-разложение(на
треугольные матрицы)",
                             command=lambda:
controller.show_frame("SecondMethod"))
        button1.pack()
        button2.pack()


class FirstMethod(tk.Frame):
    """ Страница для метода якоби """


    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        label = tk.Label(self, text="Метод Якоби", font=TITLE_FONT)
```

```
        label.pack(side="top", fill="x", pady=10)
        button = ttk.Button(self, text='Вернуться в гланое меню',
                            command=lambda:
controller.show_frame("MainMenu"))

        def jacobi_call():
            """ Построение интерфейса для ввода данных """
            n = v.get()
            for widget in self.winfo_children():
                if widget not in (
                button, n_ntr, button1, label, helping_label2,
helping_label3, helping_label4, helping_label5):
                    widget.destroy()
            if (n != "") and (int(n) in range(2, 5)):
                n = int(n)
                arr_txt = [[0] * n for i in range(0, n)]
                arr_txt1 = [[0] for i in range(0, n)]
                for i in range(0, n):
                    # helping_label = tk.Label(self, text="x0+")
                    # helping_label.place(x=60, y=130 + i * 30,
width=40, height=30)
                    for j in range(0, n):

                        if j != (n - 1):
                            helping_string = "x" + str(j) + "+"
                        elif j == (n - 1):
                            helping_string = "x" + str(j) + "="

                        helping_label = tk.Label(self,
text=helping_string)
                        arr_txt[i][j] = tk.Entry(self)
                        helping_label.place(x=100 + j * 80, y=130 + i
* 30, width=40, height=30)
                        arr_txt[i][j].place(x=60 + j * 80, y=130 + i *
30, width=40, height=30)
                    arr_txt1[i] = tk.Entry(self)
                    arr_txt1[i].place(x=60 + n * 80, y=130 + 30 * i,
width=40, height=30)
                output = tk.Text(self, bg="lightblue", font="Arial
12")
                output.place(x=470, y=130, width=300, height=300)
                helping_label4.place(x=60, y=100)
                helping_label5.place(x=470, y=435)

                def inserter(value):
                    """ Функция вывода переданных значений """
                    value1 = ""
                    for i in range(0, len(value)):
                        value1 = value1 + "x" + str(i) + " = " +
str(value[i]) + "\n" + "\n"
```

```
                    output.delete("0.0", "end")
                    output.insert("0.0", value1)

                def convert():
                    """ Получение значений и передача их функции """
                    arr_ntr = [[0] * n for i in range(0, n)]
                    arr_ntr1 = [[0] for i in range(0, n)]
                    for i in range(0, n):
                        for j in range(0, n):
                            arr_ntr[i][j] = float(arr_txt[i][j].get())
                        arr_ntr1[i] = float(arr_txt1[i].get())
                    print(*arr_ntr)
                    print(*arr_ntr1)
                    inserter(jacobi1(arr_ntr, arr_ntr1))

                btn3 = ttk.Button(self, text="Решить",
                                  command=convert)
                btn3.place(x=385, y=260, width=60, height=30)

            else:
                messagebox.showinfo("Razmernost", "Viberite
razmernost: 2 - 4")

        button1 = ttk.Button(self, text="ok",
                             command=jacobi_call)  # lambda:
messagebox.showinfo("Razmernost", "Viberite razmernost: 2 - 4"))
        v = tk.StringVar(self)
        v.set('2')
        n_ntr = ttk.OptionMenu(self, v, '2', '2', '3', '4')  #
tk.Entry(self, width=3) #sdelat combobox i kopitrnut
        button.place(x=550, y=20)
        n_ntr.place(x=330, y=50)
        button1.place(x=370, y=48)
        dop_infa = "Метод Якоби — разновидность метода простой
итерации \nдля решения системы линейных алгебраических уравнений.\n
Назван в честь Карла Густава Якоби.\nПри большом числе неизвестных
метод Гаусса \nстановится весьма сложным в плане вычислительных и
временных затрат. \nПоэтому иногда удобнее использовать приближенные
(итерационные) \nчисленные методы, метод Якоби относится к таким."
        helping_label2 = tk.Label(self, text="Выберите размерность
матрицы и нажмите OK")
        helping_label2.place(x=60, y=50)
        helping_label3 = tk.Label(self, text=dop_infa)  # TODO: vnesti
infu
        helping_label3.place(x=30, y=400)
        helping_label4 = tk.Label(self, text="Введите коэффициенты
уравнений ниже:")

        helping_label5 = tk.Label(self, text="""Окно вывода
результатов.\nB случае если ничего не выводится - коэффициенты\nзаданы
не правильно\nлибо невозможно найти корни.""")
```

```
        def destroy():
            for widget in self.winfo_children():
                if widget not in (button, n_ntr, button1, button2,
label):
                    widget.destroy()

                    # button2 = ttk.Button(self, text="Clear this
page",
                #                      command=destroy)
                # button2.place(x=500, y=100)


class SecondMethod(tk.Frame):
    """ Страница метод LU-разложения """


    def __init__(self, parent, controller):
        # self.destroy()
        tk.Frame.__init__(self, parent)
        self.controller = controller
        label = tk.Label(self, text="LU декомпозиция",
font=TITLE_FONT)
        label.pack(side="top", fill="x", pady=10)
        button = ttk.Button(self, text="Вернуться в гланое меню",
                            command=lambda:
controller.show_frame("MainMenu"))

        def lu_call():
            """ Построение интерфейса для ввода данных """
            n = v.get()
            for widget in self.winfo_children():
                if widget not in (
                button, n_ntr, button1, label, helping_label2,
helping_label3, helping_label4, helping_label5):
                    widget.destroy()
            if (n != "") and (int(n) in range(2, 6)):
                n = int(n)
                arr_txt = [[0] * n for i in range(n)]
                # arr_txt1 = [[0] * n for i in range(n)]
                arr_ntr = [[0] * n for i in range(n)]
                # arr_ntr1 = [[0] * n for i in range(n)]
                # messagebox.showinfo("Razmernost", "ok")
                for i in range(0, n):
                    for j in range(0, n):
                        arr_txt[i][j] = tk.Entry(self)
                        arr_txt[i][j].place(x=100 + j * 50, y=150 + i
* 30, width=40, height=30)
                    # arr_txt1[i] = tk.Entry(self)
                    # arr_txt1[i].place(x=100+n*50, y=100+30*i,
width=40, height=30)
```

```
                output = tk.Text(self, bg="lightblue", font="Arial 8")
                output.place(x=450, y=150, width=300, height=300)
                helping_label4.place(x=60, y=100)
                helping_label5.place(x=450, y=455)

                def inserter2(value, value2):
                    """ Функция вывода переданных значений """
                    output.delete("0.0", "end")
                    value = "U = \n" + str(value) + " \n \n"
                    value2 = "L = \n" + str(value2)
                    output.insert("0.0", value)
                    output.insert("16.0", value2)

                def convert():
                    """ Получение значений и передача их функции """
                    for i in range(0, n):
                        for j in range(0, n):
                            arr_ntr[i][j] = float(arr_txt[i][j].get())
                        # arr_ntr1[i] = float(arr_txt1[i].get())
                    L, U = lu(arr_ntr)
                    inserter2(L, U)

                btn3 = ttk.Button(self, text="Решить",
                                  command=convert)
                btn3.place(x=385, y=310, width=60, height=30)

            else:
                messagebox.showinfo("Razmernost", "Viberite
razmernost: 2 - 7")

        button1 = ttk.Button(self, text="ok",
                             command=lu_call)  # lambda:
messagebox.showinfo("Razmernost", "Viberite razmernost: 2 - 4"))
        v = tk.StringVar(self)
        v.set('2')
        n_ntr = ttk.OptionMenu(self, v, '2', '2', '3', '4', '5')
        button.place(x=550, y=20)
        n_ntr.place(x=330, y=50)
        button1.place(x=370, y=48)
        helping_label2 = tk.Label(self, text="Выберите размерность
матрицы и нажмите OK")
        helping_label2.place(x=60, y=50)
        helping_label4 = tk.Label(self, text="Введите элементы матриц
в поля ввода:")

        dop_infa = "LU-разложение (LU-декомпозиция, LU-факторизация) —
представление \nматрицы A в виде произведения двух матриц, A=LU, \nгде
L — нижняя треугольная матрица, \na U — верхняя треугольная матрица.
\nLU-разложение используется для решения \nсистем линейных уравнений,
\nобращения матриц и вычисления определителя.\nЭтот метод является
одной из разновидностей метода Гаусса."
```

```
        helping_label3 = tk.Label(self, text=dop_infa)  # TODO: vnesti
infu
        helping_label3.place(x=10, y=400)
        helping_label5 = tk.Label(self, text="""Окно вывода
результатов.\nВ случае если ничего не выводится - элементы\nматрицы
заданы не правильно\nлибо невозможно найти решения.""")

        def destroy():
            for widget in self.winfo_children():
                if widget not in (button, n_ntr, button1, button2,
label):
                    widget.destroy()

                    # button2 = ttk.Button(self, text="Clear this
page",
                #                      command=destroy)
                # button2.place(x=450, y=100)


if __name__ == "__main__":
    app = SampleApp()
    app.minsize(800, 600)
    app.maxsize(800, 600)
    app.mainloop()
```

```
# -*- coding: utf-8 -*-
"""
numpy - содержит вычислительные функции, которых нет в стандартной
библиотеке.
 Array - конвертирует список в массив;
 zeros - создать массив с заданной размерностью  и заполнить его
нулями;
 diag - извлечь диагональ массива;
 diagflat - создание массива с заданной размерностью и указаной
диагональю, и заполнить все остальное нулями;
 dot - сложение матриц.
"""

from pprint import pprint
from numpy import array, zeros, diag, diagflat, dot


def jacobi(A, b, N=1000, x=None):
    if x is None:
        x = zeros(len(A[0]))       # создаем векрот X
    pprint(x)
    print(len(A[0]))
    D = diag(A)                           # создаем D и заполняем его
диагональю из A
    R = A - diagflat(D)            # удаляем(вычитаем) диагональ(D) из
A
                                                  # A = D + R

    for i in range(N):             # цикл для вычисления корня(корней)
по формуле: x^(k+1)= (b – Rx^(k) )D^–1
        x = (b - dot(R, x)) / D

    return x


def jacobi1(a, b):
    """Вспомогательная функция для вызова основной(jacobi)"""
    a = array(a)
    b = array(b)
    #guess = array([1.0, 1.0, 1.0])
    sol = jacobi(a, b, N=1000)

    return sol


if __name__ == '__main__':
    A = array([
                [8., 4., 2.],
                [3., 5., 1.],
                [3., -2., 10.]
```

```
 ])
b = array([10., 5., 5.])
#guess = array([1.0, 1.0, 1.0])
#A = array([
#    [1., 2.],
#    [1., 2.]
#])
#b = array([3., 3.])
sol = jacobi(A, b, N=50)

print("A:")
pprint(A)

print("b:")
pprint(b)

print("x:")
pprint(sol)
```

```
# -*- coding: utf-8 -*-
"""
numpy/scipy - содержит вычислительные функции, которых нет в
стандартной библиотеке.
 Array - конвертирует список в массив;
pprint - форматированый вывод.
"""
import scipy
import pprint

def lu(a):
	n = len(a)
	L = [[0] * n for i in range(n)]
	U = [[0] * n for i in range(n)]
	for i in range(n):
		L[i][i] = 1
	for i in range(n):
		for j in range(n):
			U[i][j] = a[i,j]
	for i in range(0, n):
			for j in range(i, n):
				L[j][i] = (U[j][i])/(U[i][i])
	for k in range(1, n):
		for i in range(k-1, n):
			for j in range(i, n):
				L[j][i] = (U[j][i])/(U[i][i])
		for i in range(k, n):
			for j in range(k-1, n):
				U[i][j] = U[i][j] - L[i][k-1]*U[k-1][j]
	return L, U

if __name__ == '__main__':
	A = ([ [7, 3, -1, 2], [3, 8, 1, -4], [-1, 1, 4, -1], [2, -4, -
1, 6] ])
	A = scipy.array(A)
	L, U = lu(A)
	pprint.pprint(A)
	pprint.pprint(L)
	pprint.pprint(U)
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Результаты выполнения программы

Ввод и вывод для метода Якоби представлен на рисунке Б.1.

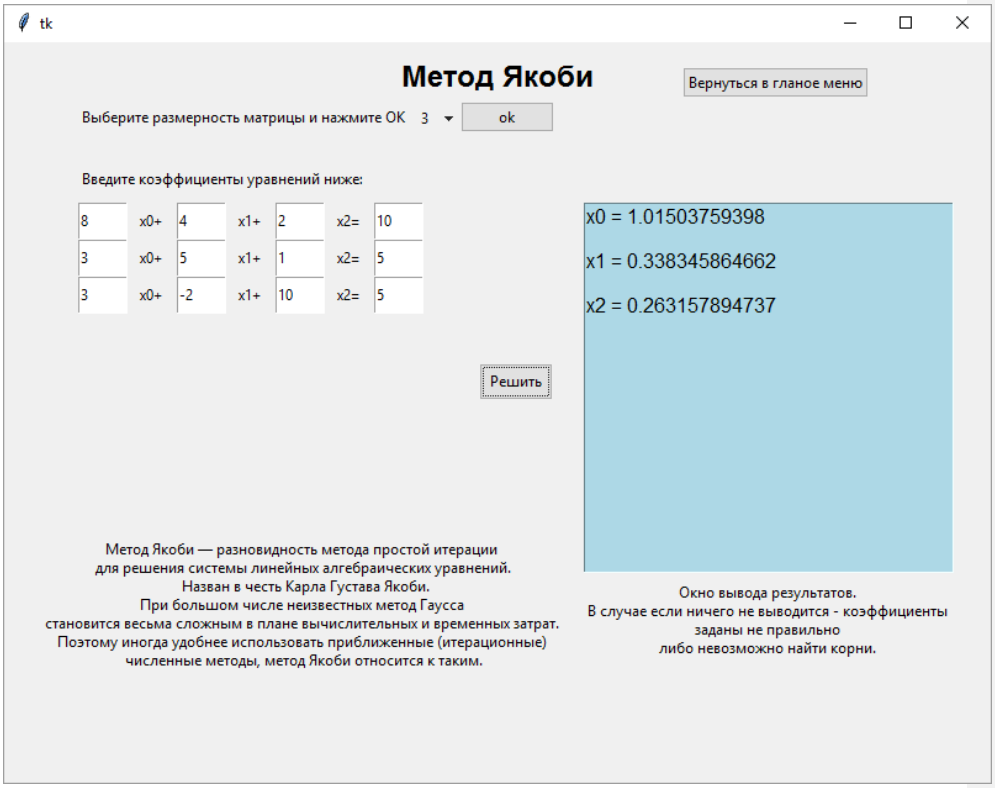


Рисунок Б.1 – Интерфейс 1-го метода

Ввод и вывод для метода Якоби представлен на рисунке Б.2.

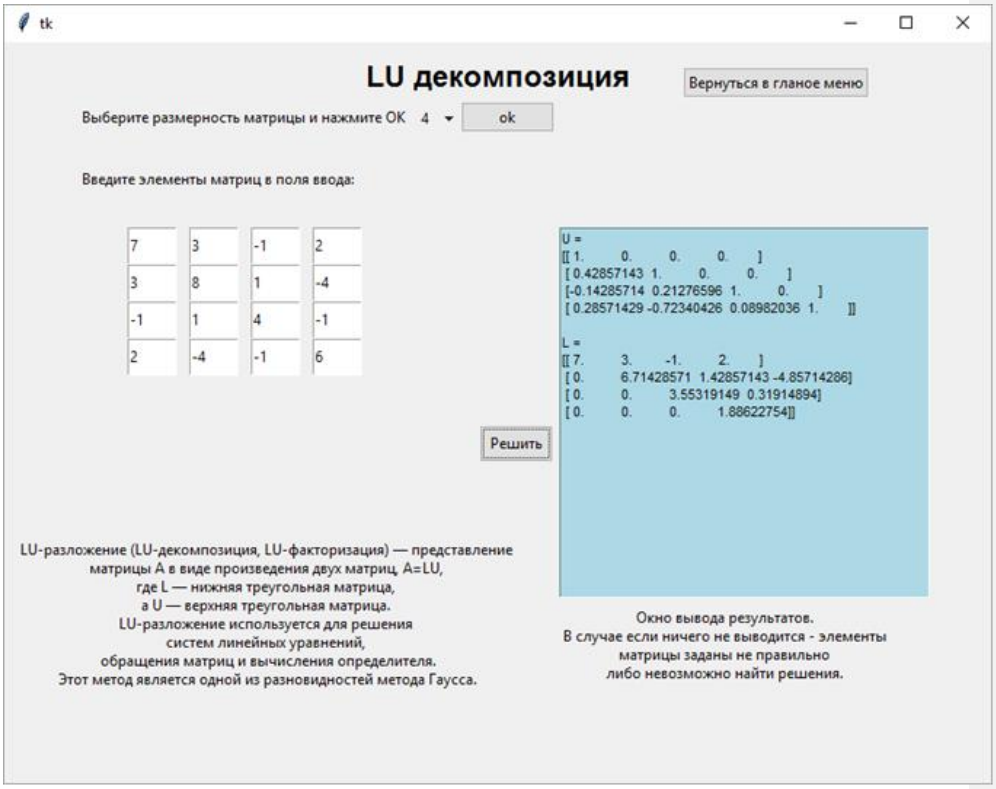


Рисунок Б.2 – Интерфейс 2-го метода

Экстремальные условия представлены на рисунке Б.3.

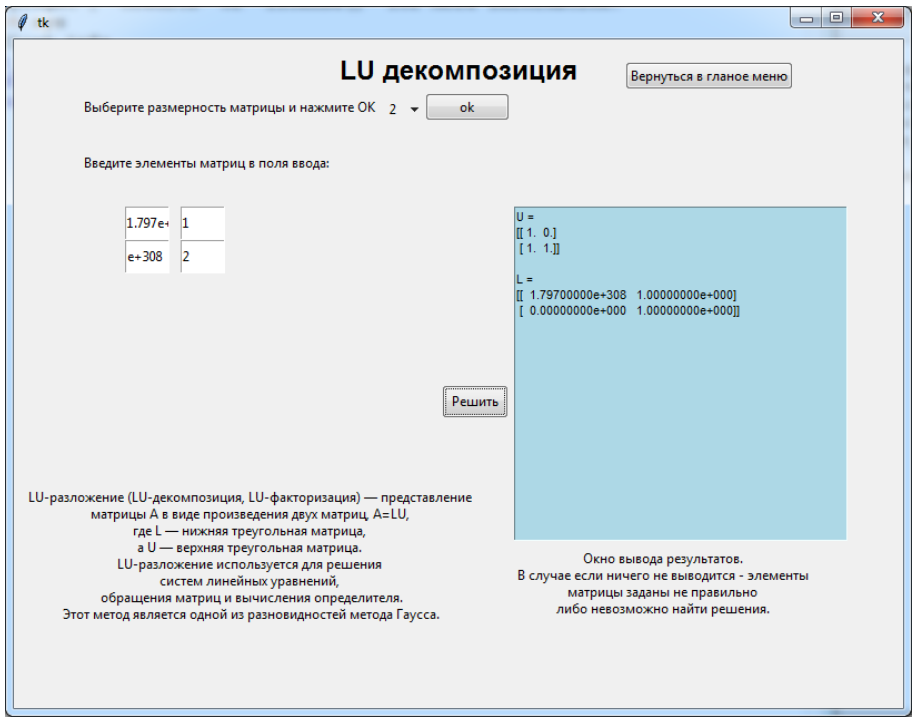


Рисунок Б.3 – Экстремальные условия.

Исключительные условия представлены на рисунке Б.4.

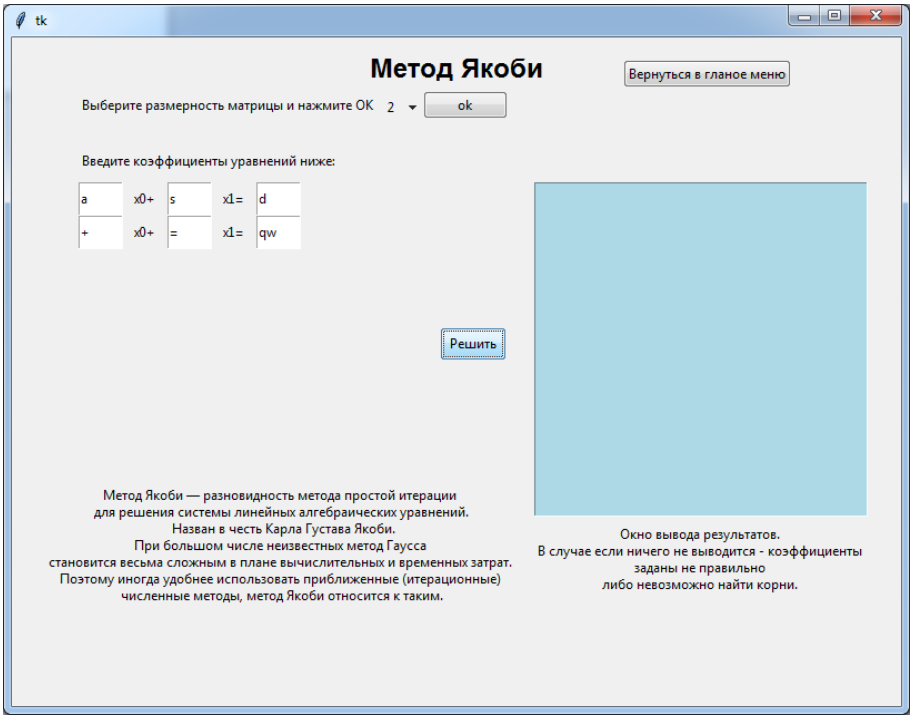


Рисунок Б.4 – Исключительные условия.