

Содержание

1. Постановка задачи.....	2
2. Разработка алгоритма решения задачи.....	3
2.1 Описание Python.....	3
2.2 PyGTK.....	4
2.3 Теория Метод Гаусса.....	5
2.3.1 Алгоритм.....	5
2.3.2 Применение и модификации.....	6
3. Листинг программы решения задания.....	7
4. Описание работы.....	13
5. Результаты выполнения работы.....	14
6 Список используемой литературы.....	15

1. Постановка задачи

Разработать программу для поиска обратной матрицы методом Гаусса в среде Python с использованием библиотеки GTK+.

2. Разработка алгоритма решения задачи.

2.1 Описание Python.

Python (англ. *Python* – питон, произносится ['paɪθ(ə)n] – *пайтон*; в русском языке распространено название *пυт́он* [1]) – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Питоне организовывается в функции и классы, которые могут объединяться в модули (которые в свою очередь могут быть объединены в пакеты).

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ [2]. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные [3]. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Питона на самом Питоне, что уменьшает затраты на изменения языка и постановку экспериментов над новыми возможностями.

Python – активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль

выполняет Cpython.

2.2 PyGTK.

PyGTK – набор Python-привязок для библиотеки графического интерфейса GTK+. PyGTK является свободным ПО и распространяется на условиях GNU LGPL. Библиотека была выбрана в качестве официального инструментария разработки для программы «Ноутбук за 100 долларов».

Начиная с версии 2.8, обёртки объектов GLib вынесены в отдельную библиотеку – PyGObject, которая должна полностью вытеснить PyGTK при использовании GTK+ версии 3.

GTK+ (сокращение от **GIMP ToolKit**) – кроссплатформенная библиотека элементов интерфейса. Наряду с Qt является одной из двух наиболее популярных на сегодняшний день библиотек для X Window System.

Будучи изначально частью графического редактора GIMP, она развилась в отдельный проект и приобрела заметную популярность. GTK+ – свободное ПО, распространяемое на условиях GNU LGPL, позволяющей создавать как свободное, так и проприетарное программное обеспечение с использованием библиотеки. GTK+ является официальной библиотекой для создания графического интерфейса проекта GNU.[4]

Собственно GTK+ состоит из двух компонентов: *GTK* и *GDK*. Первый содержит набор элементов пользовательского интерфейса, или «виджетов» (таких, как кнопка, список, поле для ввода текста и т. п.) для различных задач. GDK отвечает за вывод на экран и может использовать для этого X Window System, Linux Framebuffer, WinAPI или функции Mac OS X. Начиная с версии 2.8, GDK во многом (но не полностью) заменена на систему отрисовки векторной графики Cairo.

Помимо Cairo, GTK+ зависит от трёх библиотек – GLib, Pango и ATK, – которые разрабатываются вместе с GTK+, но могут использоваться и отдельно [4].

Внешний вид графических интерфейсов, созданных с использованием

GTK+, может конфигурироваться пользователем и/или программистом. При этом настраиваются не только цвета и шрифты, но и способ отображения различных элементов. Достигается это за счёт использования «движков» для вывода на экран. Путём подключения другого движка можно кардинальным образом менять внешний вид программ. Например, некоторые движки могут использовать псевдо-трёхмерный вид, другие – более «плоский» и т. п.

На основе GTK+ построены рабочие окружения GNOME, LXDE и Xfce. Программы, использующие GTK+, могут выполняться в других окружениях, например в KDE. Кроме того, GTK+ может работать и на операционных системах семейства Microsoft Windows и Mac OS X.

Также GTK+ выбран в качестве основной библиотеки построения пользовательских интерфейсов для Mono (порта Microsoft.NET для Linux).[5]

2.3 Теория Метод Гаусса.

Метод Га́усса – классический метод решения системы линейных алгебраических уравнений (СЛАУ). Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру) переменных, находятся все остальные переменные.

2.3.1 Алгоритм

Вычисление обратной матрицы методом Гаусса:

- 1) к матрице A приписать справа единичную матрицу E той же размерности;
- 2) путем преобразований методом Гаусса над строками расширенной матрицы $(A | E)$ матрица A приводится к единичной матрице;
- 3) в результате вычислительного процесса на месте приписанной справа матрицы E получится обратная матрица A^{-1} .

Схематично процесс нахождения обратной матрицы выглядит следующим образом: $(A | E) \rightarrow (E | A^{-1})$.

2.3.2 Применение и модификации

Помимо аналитического решения СЛАУ, метод Гаусса также применяется для:

нахождения матрицы, обратной к данной (к матрице справа приписывается единичная такого же размера, что и исходная: $[A | E]$, после чего приводится к виду единичной матрицы методом Гаусса–Жордана; в результате на месте изначальной единичной матрицы справа оказывается обратная к исходной матрица: $[E | A^{-1}]$);

определения ранга матрицы (согласно следствию из теоремы Кронекера–Капелли ранг матрицы равен числу её главных переменных);

численного решения СЛАУ в вычислительной технике (ввиду погрешности вычислений используется Метод Гаусса с выделением главного элемента, суть которого заключена в том, чтобы на каждом шаге в качестве главной переменной выбирать ту, при которой среди оставшихся после вычёркивания очередных строк и столбцов стоит максимальный по модулю коэффициент).

3. Листинг программы решения задания.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import pygtk
pygtk.require('2.0')
import gtk
import math

class Gausa:

    # Этот callback завершает программу
    def delete_event(self, widget, event, data=None):
        gtk.main_quit()
        return False

    def gauss_jordan(self, eps = 1.0/(10**10)):
        """Puts given matrix (2D array) into the Reduced
Row Echelon Form.
        Returns True if successful, False if 'm' is
singular.
        NOTE: make sure all the matrix items support
fractions! Int matrix will NOT work!
        Written by Jarno Elonen in April 2005, released
into Public Domain"""

        (h, w) = (len(self.matrix), len(self.matrix))
        for y in range(0,h):
            maxrow = y
            for y2 in range(y+1, h):          # Find max
pivot
                if abs(self.matrix[y2][y]) >
abs(self.matrix[maxrow][y]):
                    maxrow = y2
            (self.matrix[y], self.matrix[maxrow]) =
(self.matrix[maxrow], self.matrix[y])
            #if abs(m[y][y]) <= eps:          # Singular?
                #return False
            for y2 in range(y+1, h):          # Eliminate
column y
                c = self.matrix[y2][y] / self.matrix[y]
[y]
                for x in range(y, w):
                    self.matrix[y2][x] -= self.matrix[y]
[x] * c
```

```

        for y in range(h-1, 0-1, -1): # Backsubstitute
            c = self.matrix[y][y]
            for y2 in range(0,y):
                for x in range(w-1, y-1, -1):
                    self.matrix[y2][x] -=self.matrix[y]
[x] * self.matrix[y2][y] / c
                self.matrix[y][y] /= c
                for x in range(h, w): # Normalize
row y
                    self.matrix[y][x] /= c
#return True

def start_program (self):
    # Создаём новое окно
    self.window_start =
gtk.Window(gtk.WINDOW_TOPLEVEL)

    self.window_start.set_position(gtk.WIN_POS_CENTER)
    self.window_start.set_default_size(100,100)
    # Устанавливаем заголовок окна
    self.window_start.set_title("Метод Гаусса")

    # Устанавливаем обработчик для delete_event,
который немедленно
    # Завершает работу GTK.
    self.window_start.connect("delete_event",
self.delete_event)

    # Устанавливаем границу для окна
    self.window_start.set_border_width(5)

    # Создаём таблицу 3x4
    table = gtk.Table(3, 3, True)

    # Размещаем таблицу в главном окне
    self.window_start.add(table)

    # Создаём первую кнопку
    button = gtk.Button("ok")

    self.entry_1 = gtk.Entry(max=0)
    table.attach(self.entry_1, 0, 3, 1, 2,
gtk.SHRINK, gtk.SHRINK)
    self.entry_1.show()

```



```

        # По нажатию кнопки мы вызываем метод "callback"
        # с указателем на "ok" в виде аргумента
        button.connect("clicked", self.create_matrix,
None)

        # Вставляем ok в верхнюю левую ячейку таблицы
        table.attach(button, 2, 3, 2, 3, gtk.SHRINK,
gtk.SHRINK, 1, 1)
        button.show()

        label = gtk.Label("Введите размерность матрицы")
        table.attach(label, 0, 3, 0, 1)
        label.show()
        table.show()
        self.window_start.show()

def create_matrix (self, widget, data=None):
    # Создаём новое окно
    self.window_matrix =
gtk.Window(gtk.WINDOW_TOPLEVEL)

    self.window_matrix.set_position(gtk.WIN_POS_CENTER)
    self.window_matrix.set_default_size(150,150)
    # Устанавливаем заголовок окна
    self.window_matrix.set_title("Метод Гаусса")

    # Устанавливаем обработчик для delete_event,
который немедленно
    # Завершает работу GTK.
    self.window_matrix.connect("delete_event",
self.delete_event)

    # Устанавливаем границу для окна
    self.window_matrix.set_border_width(5)
    self.N = int(self.entry_1.get_text())
    # Создаём таблицу 3x4
    table = gtk.Table(self.N + 1, self.N + 2, True)

    # Размещаем таблицу в главном окне
    self.window_matrix.add(table)

    # Создаём первую кнопку
    button = gtk.Button("ok")

    # По нажатию кнопки мы вызываем метод "callback"

```

```

        # с указателем на "ok" в виде аргумента
        button.connect("clicked", self.solver, None )
        # Вставляем ok в верхнюю левую ячейку таблицы
        table.attach(button, self.N, self.N + 1, self.N
+ 1, self.N + 2, gtk.SHRINK, gtk.SHRINK, 1, 1)
        button.show()

        label      =      gtk.Label("Введите      коэффициенты
матрицы")
        table.attach(label, 0, self.N +1, 0, 1)
        label.show()

        self.entry = {}
        for i in xrange(self.N):
            for j in xrange(self.N):
                self.entry[str(i)  +  ","  +  str(j)]  =
gtk.Entry(max=0)
                table.attach(self.entry[str(i)  +  ","  +
str(j)] , i, i + 1, j +1, j +2, gtk.SHRINK, gtk.SHRINK)
                self.entry[str(i)  +  ","  +  str(j)]
.show()

        table.show()
        self.window_matrix.show()

    def solver (self, widget, data=None):
        self.matrix = []
        for i in xrange(self.N):
            row = []
            for j in xrange(self.N):
                row.append(int(self.entry[str(j)  +  ","  +
str(i)].get_text()))
            self.matrix.append(row)
        self.gauss_jordan()

        # Создаём новое окно
        self.window_matrix_res      =
gtk.Window(gtk.WINDOW_TOPLEVEL)

self.window_matrix_res.set_position(gtk.WIN_POS_CENTER)
self.window_matrix_res.set_default_size(150,150)
# Устанавливаем заголовок окна
self.window_matrix_res.set_title("Метод Гаусса")

```

```

        # Устанавливаем обработчик для delete_event,
который немедленно
        # Завершает работу GTK.
        self.window_matrix_res.connect("delete_event",
self.delete_event)

        # Устанавливаем границу для окна
self.window_matrix_res.set_border_width(5)

        # Создаём таблицу 3x4
table = gtk.Table(self.N + 1, self.N + 2, True)

        # Размещаем таблицу в главном окне
self.window_matrix_res.add(table)

        # Создаём первую кнопку
button = gtk.Button("ok")

        # По нажатии кнопки мы вызываем метод "callback"
# с указателем на "ok" в виде аргумента
button.connect("clicked", lambda w:
gtk.main_quit())
        # Вставляем ok в верхнюю левую ячейку таблицы
table.attach(button, self.N, self.N + 1, self.N
+ 1, self.N + 2, gtk.SHRINK, gtk.SHRINK, 1, 1)
button.show()

        label = gtk.Label("Результирующая матрица")
table.attach(label, 0, self.N + 1, 0, 1)
label.show()

        print self.matrix
        for i in xrange(self.N):
            for j in xrange(self.N):
                label = gtk.Label(str(self.matrix[j]
[i]))
                table.attach(label , i, i + 1, j + 1, j
+ 2, gtk.SHRINK, gtk.SHRINK)
                label.show()

        table.show()
self.window_matrix_res.show()

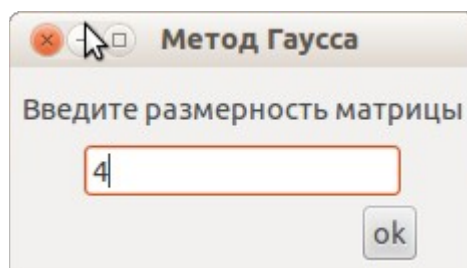
def __init__(self):
    self.start_program()

```

```
def main():  
    gtk.main()  
    return 0  
  
if __name__ == "__main__":  
    Gausa()  
    main()
```

4. Описание работы.

1. На первом этапе создается окошко с надписью «Введите матрицу», где в поле ввода необходимо ввести размер матрицы, для перехода к пункту 2 необходимо нажать «ОК».



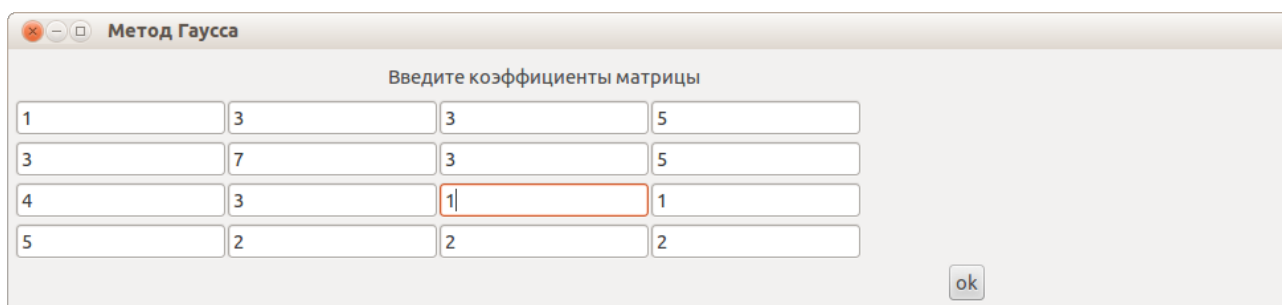
Метод Гаусса

Введите размерность матрицы

4

ok

2. На втором этапе создается окошко с надписью «Ввести коэффициенты матрицы», где в поле ввода необходимо ввести коэффициенты матрицы, для перехода к пункту 3 необходимо нажать на кнопку «ОК».



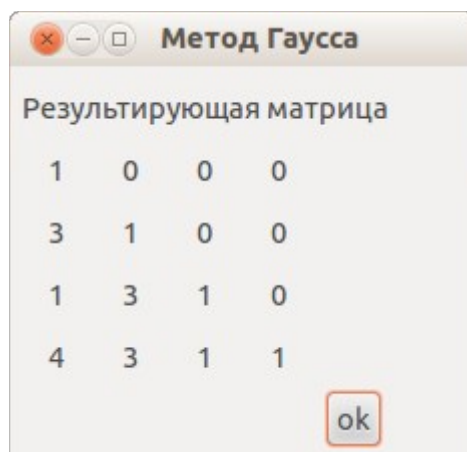
Метод Гаусса

Введите коэффициенты матрицы

1	3	3	5
3	7	3	5
4	3	1	1
5	2	2	2

ok

3. На третьем этапе создается окошко с надписью «Результирующая матрица», где выводится обратная матрица полученная по методу Гаусса, закрытия окна нажимаем на «ОК».



Метод Гаусса

Результирующая матрица

1	0	0	0
3	1	0	0
1	3	1	0
4	3	1	1

ok

5. Результаты выполнения работы.

В результате выполнения курсовой работы был осуществлен алгоритм Разработке программы для поиска обратной матрицы методом Гаусса в среде Python с использованием библиотеки GTK+. Проверка алгоритма показала адекватность его работы.

6 Список используемой литературы.

1. <http://www.xakep.ru/magazine/xa/117/088/1.asp>
2. <http://www.python.org/about/>
3. <http://www.python.org/2.5/license.html>
4. <http://mail.gnome.org/archives/gtk-devel-list/2009-September/msg00054.html>
5. <http://www.mono-project.com/GtkSharp>