第二组小组项目报告

1. 作品名称

线性方程组计算器

1. 成员介绍

信息与计算科学：郑茂森，丁宏宇

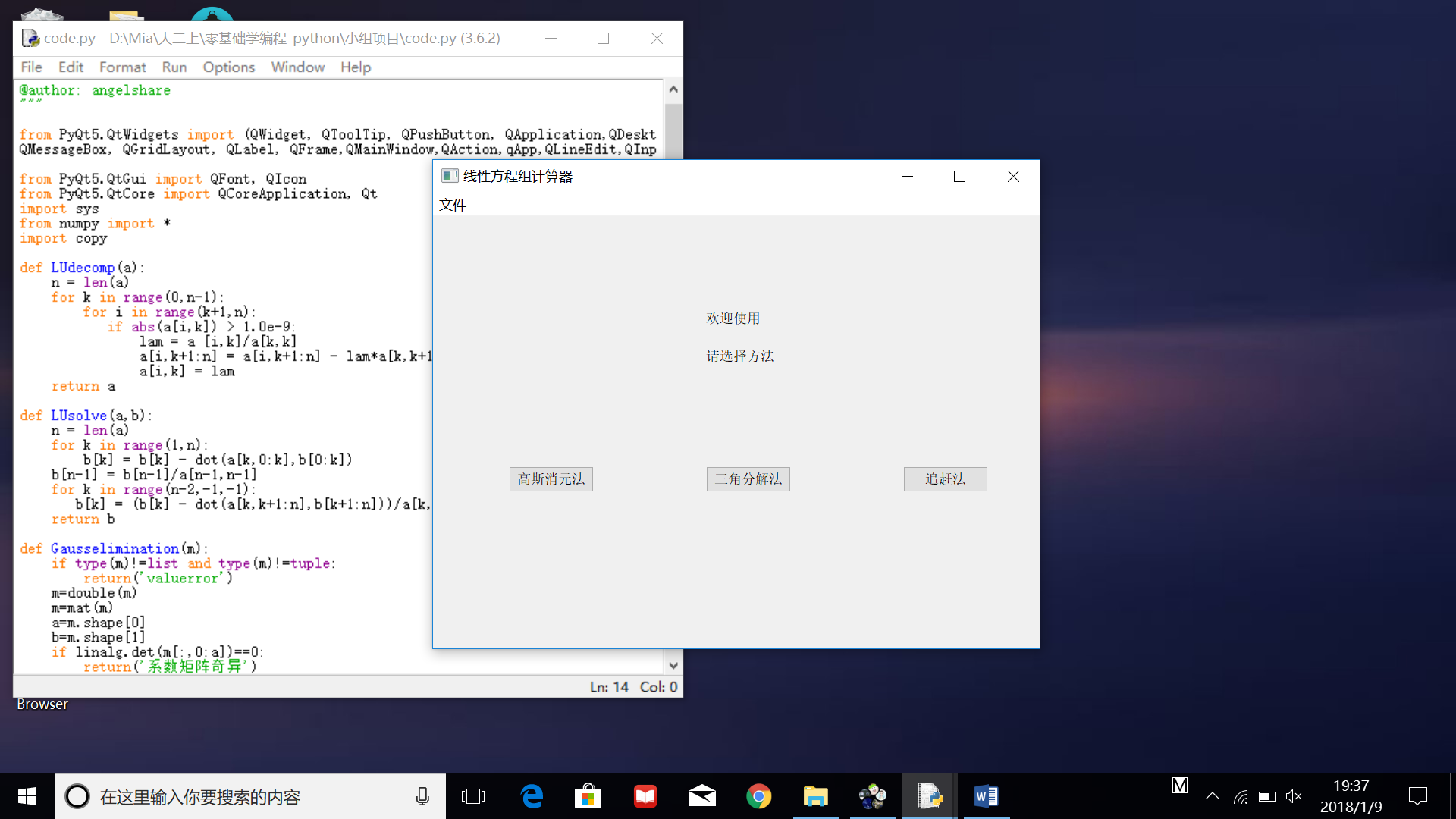
数学与应用数学：周凌一

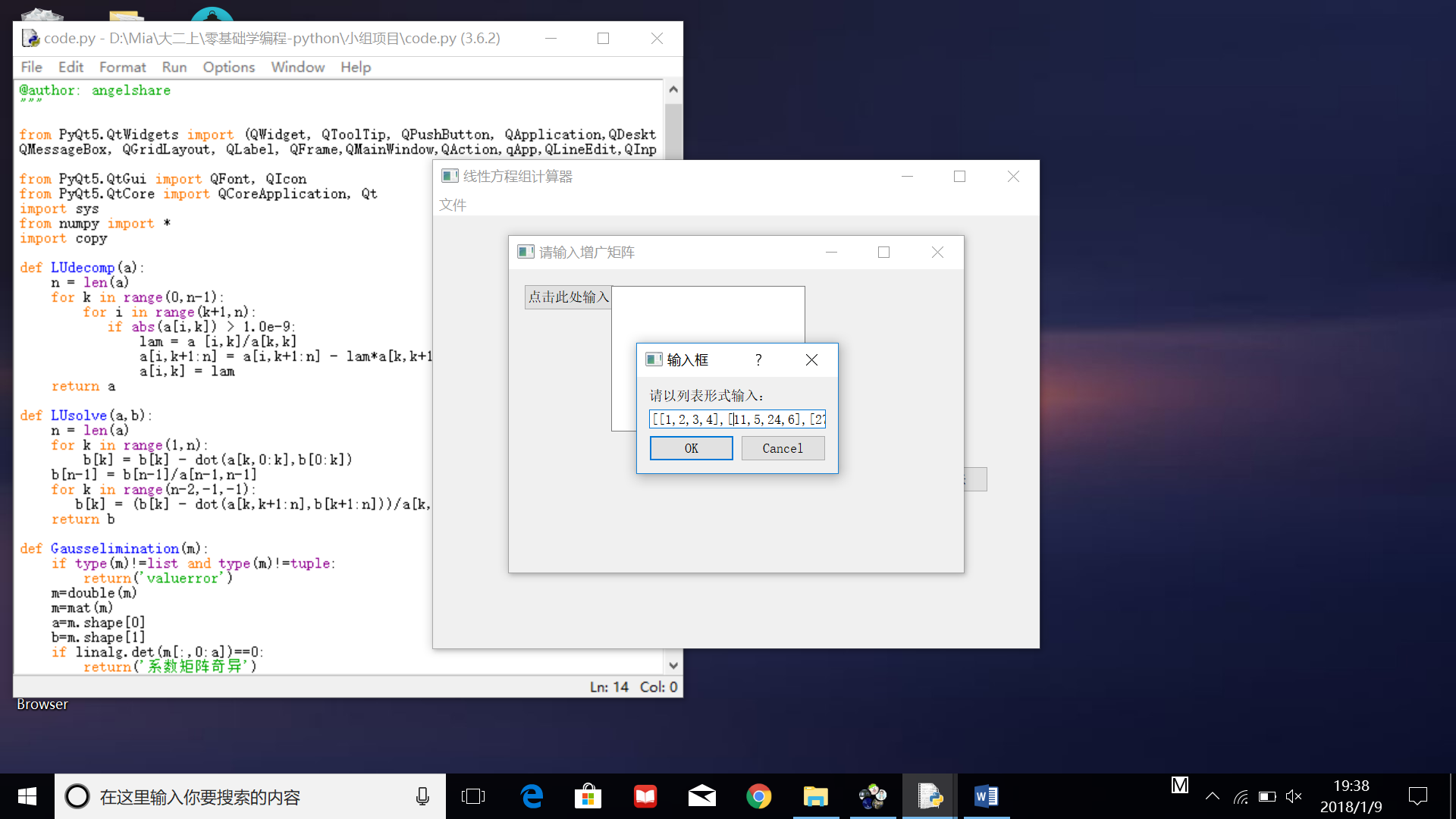
生物技术：阮朝列

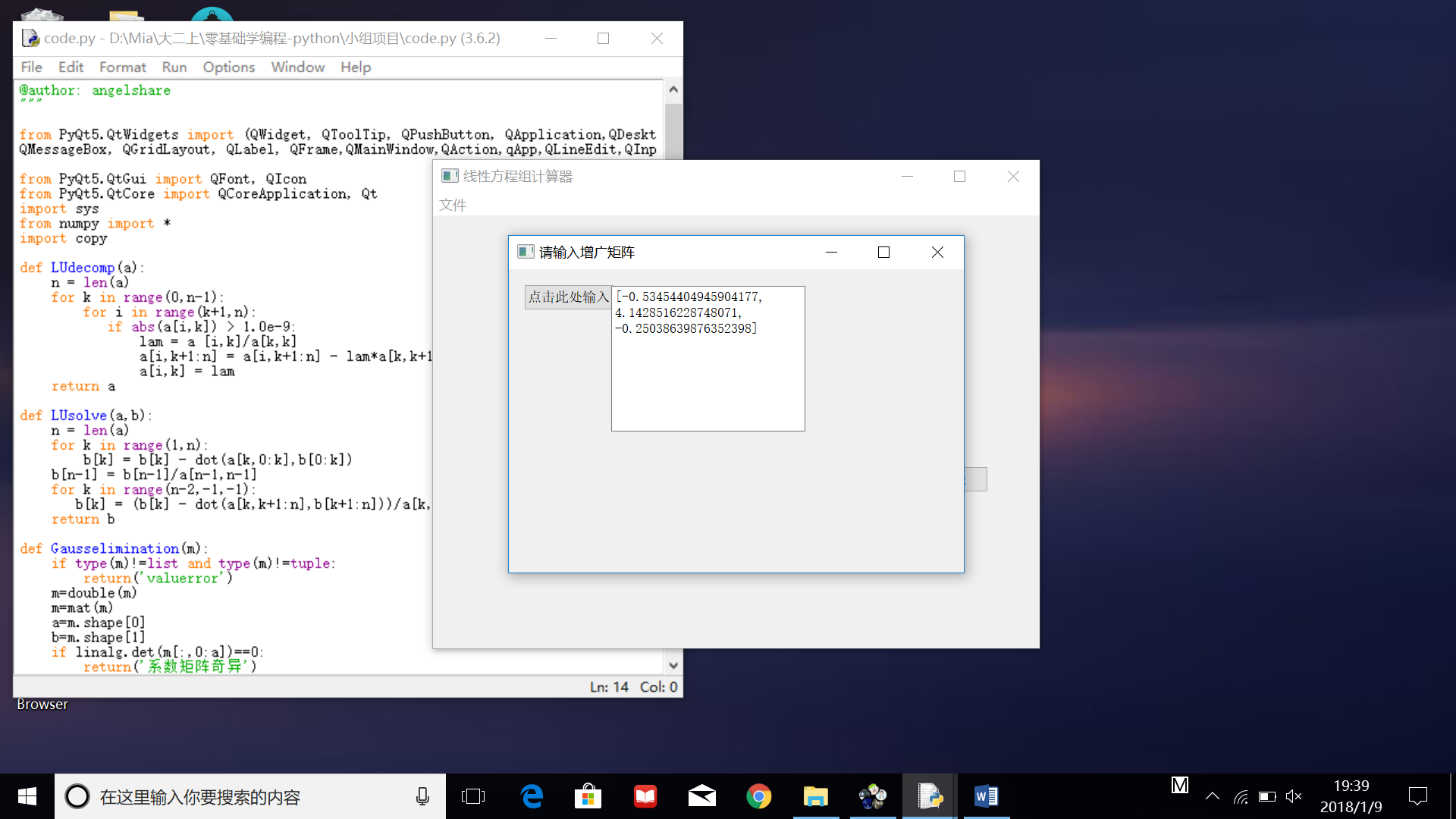
天文学：任崇阳

物理学：杨耀翔

1. 作品简介
2. 简介：作品使用PyQt5编写图形界面与用户交互。线性方程组计算器中使用了高斯消元法，三角分解法和追赶法。当矩阵为非奇异矩阵时可以使用高斯消元法。当矩阵的所有顺序主子式非0时可以使用三角分解法。追赶法适用于解三对角线形式的稀疏矩阵，因为当矩阵为稀疏矩阵时如果全部输入将占用很大内存，而且用高斯消元法所有0元素都参与运算，速度太慢，因此追赶法用克路特分解将原矩阵分解为一个下三角阵和单位上三角阵再进行运算。编写这些方法的过程中使用了numpy。
3. 应用场景：所有需要解线性方程组的场景
4. 总体设计
5. 基本思路：用图形界面让用户输入并获得输出。使用数值分析中的解线性方程组的高斯消元法，三角分解法和追赶法编写主要程序，编写过程中使用numpy。
6. 技术难点和解决方案：难点是用PyQt5编写图形界面。上网查询解决。
7. 特色和创新点
8. 特色：用数值分析中的方法，专业性比较强。
9. 创新点：用图形界面将这些方法集合起来，用户可以不懂其中的原理而直接使用这些方法解线性方程组。
10. 运行截图







1. 项目组成员的工作心得

郑茂森：线上的课程可以反复看，对基础知识的掌握有些作用。这种课程对于我们这些没有网络基础的人而言阶梯太高，一两节课也收获不了什么，不如作为扩展阅读放到线上去，让有兴趣的同学去选学。最后，我觉得python的迭代器生成器装饰器以及解析语法等比第三方库更能体现python语言。

丁宏宇：讨论区我认为需要老师更大力度的介入，至于交流也还有很大提升空间。线上课程可以让我自行分配时间，提高效率。

周凌一：线上基础知识本来就知道一些。但是因为平时编程比较少，所以见面课的知识又偏难了一点，感觉有断层。任务驱动的SPOC教学模式对我的学习有帮助。视频可以反复看，时间比较自由，可以提高效率。任务驱动目标明确。讨论区可以及时解决问题。

阮朝列：在我学习的过程中，遇到问题我首先通过百度来寻求帮助，也曾在知乎等网站求助；我认为班级群是一个寻求帮助的好地方，python课程群的老师也比较认真负责，经常问问题有问必答；此外，我还通过一些学习的网站寻求帮助，比如菜鸟教程，知乎等；我还在讨论区发帖求助，老师和同学的回复也使我比较集思广益。

任崇阳：我认为《零基础学编程—python》课程任务分割不是很合理，我觉得慕课虽然质量很高而且讲述很好，但线上线下需要大量的时间。我认为与其在前几周线下课程中对大量基础以及一些拓展的讲解还不如有老师领导先一起合作出一个项目。

杨耀翔：我觉得有所提高。多种学习模式相结合更有趣味，多种学习方式的方面不尽相同也更能让学生发现适合自己学习这门课程的学习方式。

1. 存在的问题，建议及其他要说明的情况
2. 我们的项目总体上没有什么问题。
3. 建议课程在今后开设的过程中注意线上线下内容的衔接，线上内容比较基础但是线下内容又太难了。
4. 附件：代码

# -\*- coding: utf-8 -\*-

"""

Created on Mon Oct 16 14:47:59 2017

@author: angelshare

"""

from PyQt5.QtWidgets import (QWidget, QToolTip, QPushButton, QApplication,QDesktopWidget,QLineEdit,

QMessageBox, QGridLayout, QLabel, QFrame,QMainWindow,QAction,qApp,QLineEdit,QInputDialog,QTextEdit)

from PyQt5.QtGui import QFont, QIcon

from PyQt5.QtCore import QCoreApplication, Qt

import sys

from numpy import \*

import copy

def LUdecomp(a):

n = len(a)

for k in range(0,n-1):

for i in range(k+1,n):

if abs(a[i,k]) > 1.0e-9:

lam = a [i,k]/a[k,k]

a[i,k+1:n] = a[i,k+1:n] - lam\*a[k,k+1:n]

a[i,k] = lam

return a

def LUsolve(a,b):

n = len(a)

for k in range(1,n):

b[k] = b[k] - dot(a[k,0:k],b[0:k])

b[n-1] = b[n-1]/a[n-1,n-1]

for k in range(n-2,-1,-1):

b[k] = (b[k] - dot(a[k,k+1:n],b[k+1:n]))/a[k,k]

return b

def Gausselimination(m):

if type(m)!=list and type(m)!=tuple:

return('valuerror')

m=double(m)

m=mat(m)

a=m.shape[0]

b=m.shape[1]

if linalg.det(m[:,0:a])==0:

return('系数矩阵奇异')

for j in range(a-1):

maxx=m[j,j]

k=j

for i in range(j,a):

if m[i,j]>maxx:

maxx=m[i,j]

k=i

p=copy.copy(m[j])

m[j]=m[k]

m[k]=p

for i in range(j+1,a):

m[i]=m[i]-m[j]\*(m[i,j]/maxx)

x=['']\*a

for i in range(a-1,-1,-1):

if i==a-1:

x[i]=m[a-1,a]/m[a-1,a-1]

else:

# x[i]=(m[i-1,a]-(lambda i:sum(x[j]\*m[i,j] for j in range(i+1,a))))/m[i,i]

x[i]=(m[i-1,a]-sum(x[j]\*m[i,j] for j in range(i+1,a)))/m[i,i]

#for i in range(a):

print("x(%i)=%d"%(i,x[i]))

return(x)

def inverse(m):

if type(m)!=list and type(m)!=tuple:

return('valuerror')

m=array(m)

a=m.shape[0]

b=m.shape[1]

if a!=b:

return('矩阵必须为方阵')

if linalg.det(m[:,0:a])==0:

return('矩阵奇异')

m=column\_stack([m,eye(a)])

for x in range(a):

g=m[x][x]

k=[x,x]

for i in range(x,a):

for j in range(x,a):

if m[i][j]>g:

g=m[i,j]

k=[i,j]

p=copy.copy(m[x])

m[x]=m[k[0]]

m[k[0]]=p

m[:, [x, k[1]]] = m[:, [k[1], x]]#交换列

m[x]=m[x]/g

for y in range(a):

if y!=x:

m[y]=m[y]-m[x]\*m[y,x]

return(m[:,a:2\*a])

def transposition(m):

if type(m)!=list and type(m)!=tuple:

return('valuerror')

return(list(map(list,zip(\*m))))

class wid(QMainWindow):

def \_\_init\_\_(self):

super(QMainWindow, self).\_\_init\_\_()

self.initUI()

def initUI(self):

self.resize(800, 600)

self.center()

self.setWindowTitle('线性方程组计算器')

self.setWindowIcon(QIcon('icon.jpg'))

lbl1=QLabel('欢迎使用',self)

lbl1.move(360,150)

lbl2=QLabel('请选择方法',self)

lbl2.move(360,200)

but1 = QPushButton('高斯消元法', self)

but1.resize(but1.sizeHint())

but1.move(100, 360)

but1.setObjectName("but1")

but1.clicked.connect(self.skip2)

but2 = QPushButton('三角分解法',self)

but2.resize(but2.sizeHint())

but2.move(360, 360)

but2.setObjectName("but2")

but2.clicked.connect(self.skip2)

but3 = QPushButton('追赶法',self)

but3.resize(but3.sizeHint())

but3.move(620, 360)

but3.setObjectName("but3")

but3.clicked.connect(self.skip2)

exit\_menu = QAction(QIcon(r"1.ico"), "退出", self)

exit\_menu.setShortcut("Ctrl+Q")

exit\_menu.setStatusTip("退出程序")

exit\_menu.triggered.connect(qApp.quit)

menubar = self.menuBar()

file = menubar.addMenu("文件")

file.addAction(exit\_menu)

def center(self):

qr = self.frameGeometry()

cp = QDesktopWidget().availableGeometry().center()

qr.moveCenter(cp)

self.move(qr.topLeft())

def closeEvent(self, event):

reply = QMessageBox.question(self,'确认退出','你确定要退出么？',

QMessageBox.Yes,

QMessageBox.No)

if reply == QMessageBox.Yes:

event.accept()

else:

event.ignore()

def keyPressEvent(self, event):

if event.key() == Qt.Key\_Escape:

reply = QMessageBox.question(self,'确认退出','你确定要退出么？',

QMessageBox.Yes,

QMessageBox.No)

if reply == QMessageBox.Yes:

self.close()

else:

event.ignore()

def skip2(self):

self.haoN=InputDialog()

self.haoN.show()

class InputDialog(QWidget):

def \_\_init\_\_(self):

super(InputDialog, self).\_\_init\_\_()

self.setWindowIcon(QIcon('icon.jpg'))

self.setWindowTitle("请输入增广矩阵")

self.resize(600, 400)

self.center()

self.button = QPushButton("点击此处输入", self)

self.button.setFocusPolicy(Qt.NoFocus)

self.button.move(20, 20)

self.button.clicked.connect(self.show\_dialog)

self.setFocus()

self.label = QTextEdit(self)

self.label.move(135, 22)

def center(self):

qr = self.frameGeometry()

cp = QDesktopWidget().availableGeometry().center()

qr.moveCenter(cp)

self.move(qr.topLeft())

def show\_dialog(self):

text, ok = QInputDialog.getText(self, "输入框", "请以列表形式输入：")

if ok:

text=Gausselimination(eval(text))

self.label.setText(str(text))

class wid2(QMainWindow):

def \_\_init\_\_(self):

super(QMainWindow, self).\_\_init\_\_()

self.initUI()

def initUI(self):

self.resize(800, 600)

self.center()

self.setWindowTitle('线性方程组计算器')

self.setWindowIcon(QIcon('icon.jpg'))

exit\_menu = QAction(QIcon(r"1.ico"), "退出", self)

exit\_menu.setShortcut("Ctrl+Q")

exit\_menu.setStatusTip("退出程序")

exit\_menu.triggered.connect(qApp.quit)

menubar = self.menuBar()

file = menubar.addMenu("文件")

file.addAction(exit\_menu)

def center(self):

qr = self.frameGeometry()

cp = QDesktopWidget().availableGeometry().center()

qr.moveCenter(cp)

self.move(qr.topLeft())

def closeEvent(self, event):

reply = QMessageBox.question(self,'确认退出','你确定要退出么？',

QMessageBox.Yes,

QMessageBox.No)

if reply == QMessageBox.Yes:

event.accept()

else:

event.ignore()

def keyPressEvent(self, event):

if event.key() == Qt.Key\_Escape:

reply = QMessageBox.question(self,'确认退出','你确定要退出么？',

QMessageBox.Yes,

QMessageBox.No)

if reply == QMessageBox.Yes:

self.close()

else:

event.ignore()

class wid3(QMainWindow):

def \_\_init\_\_(self):

super(QMainWindow, self).\_\_init\_\_()

self.initUI()

def initUI(self):

self.resize(800, 600)

self.center()

self.setWindowTitle('结果')

self.setWindowIcon(QIcon('icon.jpg'))

exit\_menu = QAction(QIcon(r"1.ico"), "退出", self)

exit\_menu.setShortcut("Ctrl+Q")

exit\_menu.setStatusTip("退出程序")

exit\_menu.triggered.connect(qApp.quit)

menubar = self.menuBar()

file = menubar.addMenu("文件")

file.addAction(exit\_menu)

def center(self):

qr = self.frameGeometry()

cp = QDesktopWidget().availableGeometry().center()

qr.moveCenter(cp)

self.move(qr.topLeft())

def closeEvent(self, event):

reply = QMessageBox.question(self,'确认退出','你确定要退出么？',

QMessageBox.Yes,

QMessageBox.No)

if reply == QMessageBox.Yes:

event.accept()

else:

event.ignore()

def keyPressEvent(self, event):

if event.key() == Qt.Key\_Escape:

reply = QMessageBox.question(self,'确认退出','你确定要退出么？',

QMessageBox.Yes,

QMessageBox.No)

if reply == QMessageBox.Yes:

self.close()

else:

event.ignore()

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = wid()

QToolTip.setFont(QFont('SansSerif', 10))

ex.show()

sys.exit(app.exec\_())