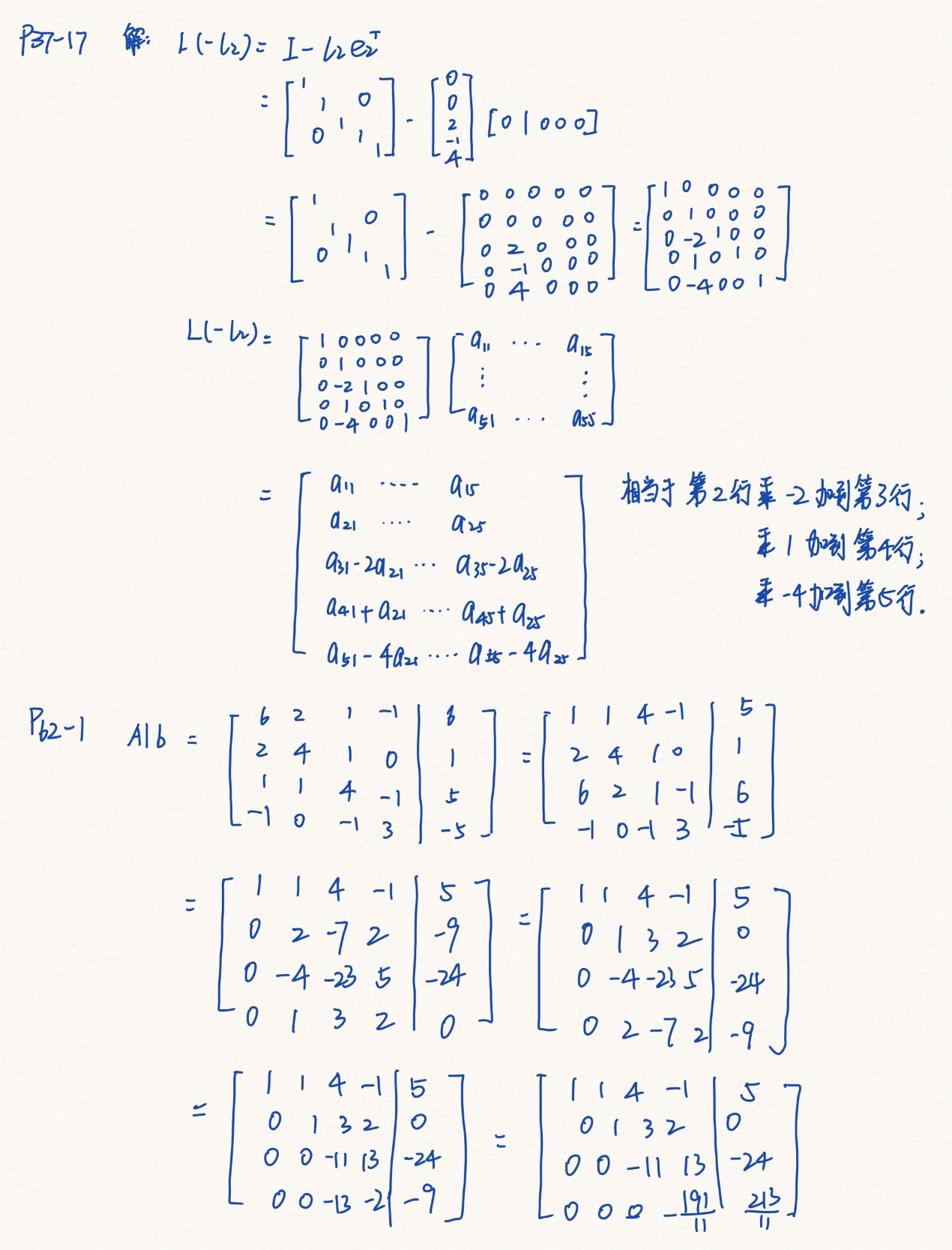
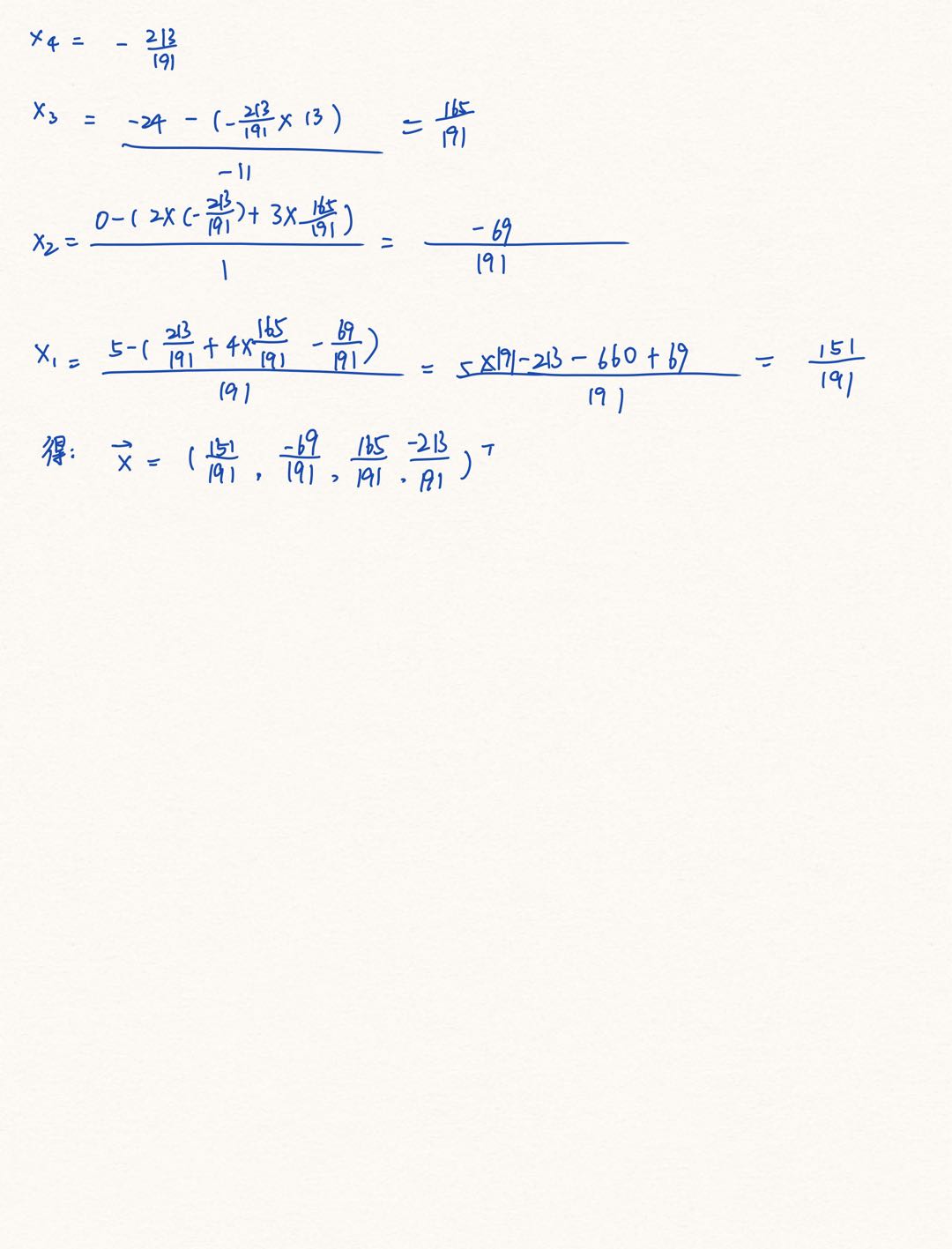
一、





二、

高斯消去程序：

# 高斯消去64

def GaussianSolve(A, b):

n = len(b)

Ab = np.c\_[Mat2, b]

for i in range(n):

a = Ab[i, i]

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

k = -Ab[j, i] / a

Ab[j:j + 1] += k \* Ab[i:i + 1]

x = np.zeros(n)

x[n - 1] = Ab[n - 1, n] / Ab[n - 1, n - 1]

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

x[i] = Ab[i, n]

for j in range(n - 1, i, -1):

x[i] -= x[j] \* Ab[i, j]

x[i] /= Ab[i, i]

return x

n = 4

Mat2 = geneSquaMat(n)

detTure = li.det(Mat2)

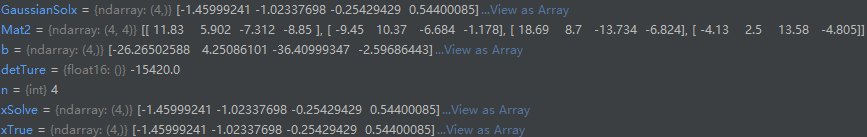
xTrue = np.random.randn(n)

b = np.inner(Mat2, xTrue)

xSolve = li.solve(Mat2, b)

GaussianSolx = GaussianSolve(Mat2, b)

结果：



编写的程序得出的高斯消元结果(GaussianSolx)和linalg包的solve函数(xSolve)以及真实值(xTrue)一致。

(1)

16位float高斯消元函数：

def GaussianSolve(A, b):

n = len(b)

Ab = np.c\_[Mat2, b]

for i in range(n):

a = Ab[i, i]

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

k = np.float16(-Ab[j, i] / a)

Ab[j:j + 1] += np.float16(k \* Ab[i:i + 1])

x = np.zeros(n)

x[n - 1] = np.float16(Ab[n - 1, n] / Ab[n - 1, n - 1])

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

x[i] = Ab[i, n]

for j in range(n - 1, i, -1):

x[i] -= np.float16(x[j] \* Ab[i, j])

x[i] /= np.float16(Ab[i, i])

return x

测试程序段：

n = 4

Mat2 = np.float16(geneSquaMat(n))

detTure = np.float16(li.det(Mat2))

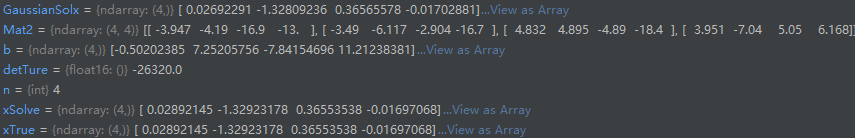
xTrue = np.random.randn(n)

b = np.inner(Mat2, xTrue)

xSolve = li.solve(Mat2, b)

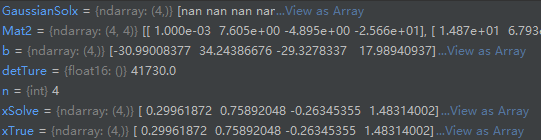
GaussianSolx = GaussianSolve(Mat2, b)

结果为：



可以看出，计算的结果和真实值及包函数的结果产生了一定的误差。

(2)将a11改成0.001：



出现NaN结果。

分析：

(1) 高斯消元方法会用到很多的乘除步骤，这些乘除步骤在计算机程序中会产生一定的误差，当计算机浮点数位数不够时，误差加大，导致结果出现了明显的误差。

(2) 消元时，方阵对角线的元素作为分母作除法，当它们特别小的时候，会导致被消元的数同一行的其他数特别大，如下图，消元比例k = -11930，再和其他元素相乘，超出16位浮点数的表示范围，出现了inf的值，再加大会出现NaN。

