Лабораторная работа №1

Вариант 3

```
# перемножение матриц
def Multiple(a, b, N, M, K):
  if (K > 1):

    c = [[0] * K for i in range(N)]
    for i in range (N):
      for j in range (K):
        c[i][j] = 0
        for l in range(M):
          s = float(a[i][l] * b[l][j])
          c[i][j] += s
    return c
  else:
    c = [0] * N
    for i in range (N):
      for j in range (K):
        c[i] = 0
        x = 0
        for l in range(M):
          x = x + float(a[i][l] * b[l][j])
        c[i] = x
    return c

# поиск обратной матрицы
def GetA_1(b):
  n = len(b)
  a = [[0] * (2 * n) for i in range(n)]
  for i in range(n):
    for j in range(n):
      a[i][j] = b[i][j]
    a[i][n + i] = 1
  # приведение левой части к верхнетреугольному виду
  for i in range(n - 1):
    for j in range(i+1, n):
      if (a[i][i] != 0):
        div = -a[i][j]/a[i][i]
        for l in range(i, 2 * n):
          a[j][l] += div * a[i][l]
  # приведение левой части к диагоальному виду

  for i in range(n - 1, -1, -1):
      for l in range (i - 1, -1, -1):
        div = - a[l][i]/a[i][i]
        a[l][i] = 0
        for j in range (n, 2* n):
```

```
          a[l][j] += a[i][j] * div
  # приведение левой части к единичной матрице, копирование в b правой
части,
  # которая будет обратной матрицей к заданной
  b = [[0] * n for i in range(n)]
  for i in range(n):
    div = a[i][i]
    for j in range(n, 2 * n):
      a[i][j] /= div
      if (j >= n):
        b[i][j - n] = a[i][j]
  return b

# поиск нормы матрицы
def GetNorm(a):
  n = len(a)
  norm = 0
  for i in range(n):
    cur = 0
    for j in range(n):
      cur += abs(a[i][j])
    norm = max(cur, norm)
  return norm

# поиск LUP разложения (по столбцу)
def GetLUP(A):
  n = len(A)
  U = [[0] * n for i in range(n)]
  for i in range(n):
    for j in range(n):
      U[i][j] = A[i][j]
  L = [[0] * n for i in range(n)]
  p_was = []
  p_now = []
  for i in range(n):
    L[i][i] = 1
  for i in range(n):
    max_element = abs(U[i][i])
    index = i
    for j in range(i + 1, n):
      if (abs(U[j][i]) > max_element):
        max_element, index = abs(U[j][i]), j
    if (max_element == 0):
      break
    p_was.append(i)
    p_now.append(index)
    for j in range(n):
      U[i][j], U[index][j] = U[index][j], U[i][j]
    for j in range(i + 1, n):
      L[j][i] = U[j][i] / U[i][i]
      for l in range(i, n):
```

```
        U[j][l] -= L[j][i] * U[i][l]
    p = []
    for i in range(n):
      p.append(i)
    for j in range(n):
        ii = int(p_was[j])
        jj = int(p_now[j])
        p[ii], p[jj] = p[jj], p[ii]

    for i in range(n):
      for j in range(i + 1, n):
        ii = int(p_was[j])
        jj = int(p_now[j])
        L[ii][i], L[jj][i] = L[jj][i], L[ii][i]
    return L, U, p

# поиск решения через LUP
def FindSolutionLUP(L, U, P, b):
    N = len(L)
    b1 = [0] * N
    for i in range(N):
      b1[i] = b[P[i]]
    solution = []
    solution.append(b1[0] / L[0][0])
    for i in range(1, N):
      left = 0
      for j in range(i):
        left += solution[j] * L[i][j]
      solution.append((b1[i] - left) / L[i][i])
    i = N - 1
    result = [0] * N
    while(i >= 0):
      left = 0
      for j in range(N):
        left += result[j] * U[i][j]
      result[i] = (solution[i] - left) / U[i][i]
      i-=1
    return result

# поиск решения методом Гаусса (по столбцу)
def FindSolutionGauss(A, d):
    n = len(A)
    G = [[0] * n for i in range(n)]
    b = [0] * n
    for i in range(n):
      b[i] = d[i]
      for j in range(n):
        G[i][j] = A[i][j]

    for i in range(n):
```

```
    max_element = abs(G[i][i])
    index = i
    for j in range(i + 1, n):
      if (abs(G[j][i]) > max_element):
        max_element, index = abs(G[j][i]), j
    if (max_element == 0):
      break
    for j in range(n):
      G[i][j], G[index][j] = G[index][j], G[i][j]
    b[i], b[index] = b[index], b[i]
    for j in range(i + 1, n):
      div = G[j][i] / G[i][i]
      b[j] = b[j] - b[i] * div
      for l in range(i, n):
        G[j][l] -= div * G[i][l]
  i = n - 1
  result = [0] * n
  while(i >= 0):
    left = 0
    for j in range(i + 1, n):
      left += result[j] * G[i][j]
    result[i] = (b[i] - left) / G[i][i]
    i-=1
  return result

# решение методом LDLT
def FindSolutionLDLT(L, D, LT, b):
  N = len(L)
  LD = Multiple(L, D, N, N, N)
  y = []
  y.append(b[0] / LD[0][0])
  for i in range(1, N):
    left = 0
    for j in range(i):
      left += y[j] * LD[i][j]
    y.append((b[i] - left) / LD[i][i])

  i = N - 1
  solution = [0] * N
  while(i >= 0):
    left = 0
    for j in range(N):
      left += solution[j] * LT[i][j]
    solution[i] = (y[i] - left) / LT[i][i]
    i-=1
  return solution

def FindSolutionSQRT(A, b):
  N = len(A)
  SQRT = [[0] * N for i in range(N)]
```

```
  for i in range(N):
    for j in range(N):
      SQRT[i][j] = A[i][j]

  for i in range(N):
    for j in range(i+1, N):
      div = SQRT[j][i] / SQRT[i][i]
      for l in range(N):
        SQRT[j][l] -= div * SQRT[i][l]
    for j in range(i + 1, N):
      SQRT[i][j] /= SQRT[i][i]


  for i in range(N):
    for j in range(i+1, N):
      SQRT[i][j] *= abs(SQRT[i][i]) ** 0.5
      SQRT[i][j] *= abs(SQRT[i][i])/SQRT[i][i]
    SQRT[i][i] /= abs(SQRT[i][i]) ** 0.5

  y = []
  for i in range(0, N):
    left = 0
    for j in range(i):
      left += y[j] * SQRT[j][i]
    y.append((b[i] - left) / SQRT[i][i])

  for i in range(0, N):
    for j in range(i + 1, N):
      SQRT[i][j] *= abs(SQRT[i][i]) / SQRT[i][i]
    SQRT[i][i] *= abs(SQRT[i][i]) / SQRT[i][i]

  i = N - 1
  solution = [0] * N
  while(i >= 0):
    left = 0
    for j in range(i + 1, N):
      left += solution[j] * SQRT[i][j]
    solution[i] = (y[i] - left) / SQRT[i][i]
    i-=1
  return solution

# построение LDLT разложения
def FindLDLT(A):
  N = len(A)
  U = [[0] * N for i in range(N)]
  L = [[0] * N for i in range(N)]
  D = [[0] * N for i in range(N)]
  LT = [[0] * N for i in range(N)]
  for i in range(N):
```

```
    for j in range(N):
      U[i][j] = A[i][j]

  for i in range(N):
    for j in range(i + 1, N):
      L[j][i] = U[j][i] / U[i][i]
      LT[i][j] = L[j][i]
      for l in range(i, N):
        U[j][l] -= L[j][i] * U[i][l]

  for i in range(N):
    #L[i][i] = abs(A[i][j]) ** 0.5
    #LT[i][i] = abs(A[i][j]) ** 0.5
    #D[i][i] = A[i][j] / abs(A[i][j])
    D[i][i] = U[i][i]
    L[i][i] = 1
    LT[i][i] = 1
  return L, D, LT, U

# вычисление нормы вектора
def GetVectorNorm(v):
  mx = 0
  n = len(v)
  for i in range(n):
    mx = max(mx, abs(v[i]))
  return mx

# метод релаксаци
def Relaxation(A, b, w, eps):
  N = len(A)
  x_i = [0] * N
  x_i1 = [0] * N
  r = [0] * N
  A1 = [[0] * N for i in range(N)]
  b1 = [0] * N
  go = 1
  c = 0
  val = []
  while(go):
    c+=1
    for i in range(N):
      before_i = 0
      after_i = 0
      for j in range(i):
        A1[i][j] = -A[i][j] * (w / A[i][i])
        before_i += A1[i][j] * x_i1[j]
      for j in range(i + 1, N):
        A1[i][j] = -A[i][j] * (w / A[i][i])
        after_i += A1[i][j] * x_i[j]
      b1[i] = b[i] * w / A[i][i]
```

```
      x_i1[i] = before_i + after_i + b1[i] - x_i[i] * (w - 1)
    for i in range(N):
      r[i] = x_i1[i] - x_i[i]
    for i in range(N):
      x_i[i] = x_i1[i]
    norm = GetVectorNorm(r)
    val.append(norm)
    if (norm < eps):
       break
  return x_i1, c, val
```

Задание 1. Заполнить верхний треугольник матрицы A размером 256×256 рациональными случайными числами из полуинтервала $\left[-2^{3/4}, 2^{3/4}\right]$.

Нижний треугольник матрицы A заполнить таким образом, чтобы выполнялось $A = A^T$. Диагональные элементы получить из формулы $a_{ii} = 1 + \sum_{j \neq i} |a_{ij}|$.

```
V = 3
N = 256
M = 3/4
Lborder = - (2**M)
Rborder = (2**M)
A = [[0] * N for i in range(N)]
for i in range(0, N):
    A[i][i] = 1
    for j in range(i):
      A[i][i] += abs(A[i][j])
    for j in range(i + 1, N):
      A[i][j] = random.uniform(Lborder, Rborder)
      A[j][i] = A[i][j]
      A[i][i] += abs(A[i][j])
```

Задание 2a. Заполнить вектор y длиной соответствующей размеру матрицы A рациональными случайными числами из полуинтервала ¿.

```
y = [[0] * 1 for i in range(N)]
for i in range(N):
  y[i][0] = random.uniform(Lborder, Rborder)
```

Задание 2b. Умножив матрицу A на вектор y получить вектор правой части b. Таким образом имеем СЛАУ $Ax = b$, точным решением которой является вектор y.

```
b = Multiple(A, y, int(N), int(N), int(1))
```

Задание 3. Найти число обусловленности матрицы A, вычислив A^{-1} методом Гаусса-Жордана, в качестве нормы матрицы выбрать кубическую норму.

Вычислим обратную матрицу:

```
start_time = time.time()
A_1 = GetA_1(A)
find_inverse_matrix_time = (time.time() - start_time)
print("Время нахождения обратной матрицы: ", find_inverse_matrix_time)
```

```
Время нахождения обратной матрицы:  5.0836405754089355
```

Посчитаем норму матриц:

```
norm = GetNorm(A)
norm_1 = GetNorm(A_1)
```

Число обусловленности равно $\varkappa(A)=\|A^{-1}\|\|A\|$.

```
EA = norm * norm_1
print("Число обусловленности:", EA)
```

```
Число обусловленности: 4.58742154113254
```

Задание 4. Решить СЛАУ $Ax=b$ методом Гаусса с выбором главного элемента по столбцу.

```
start_time = time.time()
resultGauss = FindSolutionGauss(A, b)
print("Время решения: ", (time.time() - start_time))
print("Решение, полученное методом Гаусса: ", resultGauss)
```

```
Время решения:  1.3230793476104736
Решение, полученное методом Гаусса:  [0.6566520880736906, -
0.23015396915304306, 0.8719050790323561, 0.03889790878458688, -
0.07604230469555953, 0.7721991480879276, -0.938474607699839,
1.1763115132406285, -1.3507796111277872, -1.6569177393456582, -
0.8041701327377628, -0.8746041525873355, -1.094815859077489, -
1.5740746421332341, -1.5111472965494803, -0.11258947259761329, -
0.5577354403269896, 0.17795955713729106, -1.4793268114463864, -
1.5093983615762903, -0.703355267133363, 0.7469712701273155,
0.1786099580423231, -0.9479902679879054, 0.21053898437820628, -
1.6046163694620912, 0.09121873608015335, -0.2758297036696052, -
0.7300422645917519, -0.9805127995707497, -0.6458456790615861, -
0.6076642701604529, 0.33428081769441303, 1.2736725600124426,
1.0068775124101998, 0.004215003943413472, -1.5789447043023552,
1.0354522683459089, -1.673374078851239, -0.21883758615992088,
1.5063936609113748, -0.9283957151128618, 0.7714893395480988,
0.6819383237977342, 1.3699399604611846, -0.24655231785576096, -
0.5003655030263529, -1.4639935215738134, -0.9078115059609079,
0.4004934506652631, -0.9598823226045753, 1.2885652185523095,
1.6736365609380057, -1.2396873735871856, 0.4387477806224906,
0.12476954356174651, 0.29500581150982347, 0.5279432244008171,
0.6463619700492648, 0.501388108994809, 0.27203464366399027, -
0.4794930863639286, -0.942040520824847, -0.7622833817080795, -
```

0.5036210640529875, -1.2865881300685473, -1.262727516962768, -
1.4901062128304623, -1.0555449667740457, 0.9050305997444769,
1.1979169001504422, -1.0719694064816874, -0.4398891064390027, -
0.8271635839058393, 1.4964656565384744, 0.5630264307169506, -
1.6088472486336705, 1.1864870399915186, -0.3505377902453475,
1.1909654190601466, 1.0794328689664998, -0.0453256001795472,
0.08350639470739314, -1.3251951746491872, 0.7029877673239676, -
0.9170620217096687, 1.358369044167818, 0.868799801870162, -
0.8076310098916317, -1.4581446472261512, 1.0455048471423474,
0.3522184766305132, 0.6172514400289995, -1.650036304070174,
0.5199579627623377, -0.3592840823260098, -0.5021789039155587,
1.5057492417995686, 0.9606489566335502, -0.5352524353466385, -
0.28135878372290296, -1.1531343515794878, 1.26560995944166,
0.6741477856125327, -0.8556905246580938, 0.9201219232495522,
0.2505555889794707, -1.148632651867478, 0.08845349720963622,
0.48278949229926066, 1.1718895557632538, -0.8582132233710053,
0.8355045742968255, -0.0013565083686086594, -0.8062125167679918, -
1.0109302810108995, 0.7554063756923801, -0.25552032663468727,
1.520223134249328, 1.5479529402765286, 0.3037491463722284, -
0.24880972641921828, -0.5318071662735092, 0.2624026093691663,
0.17891724585872218, 1.3188738405427678, -1.6079924072429999,
1.5739463276075139, 1.4672874254194497, 0.09154033535685004, -
1.5899849535118264, -0.1500190218530594, 0.660120724971521,
1.5387219541727917, 1.2751700352334123, -1.2872083927193874, -
1.4057165081812737, 0.8516600641288808, 1.2240979993347758,
0.43689043395159594, 1.584175247569991, -0.9885212448641069,
1.6080889200481936, -0.3908623951274675, 1.42874817026728, -
1.2130417569955638, 0.2771592244571677, -0.6963307104299437, -
1.2966121311418282, -0.5328686379866259, 1.1678567671302602,
0.6625521235273333, 0.7626892011585754, -0.6770712985654163, -
1.1080400762661289, 0.46621565170007107, -0.6383378249033106,
1.1897609160322407, 0.6380274709985785, 0.21023759931197716, -
0.6625288294284901, 0.5140192138338582, 0.9856140306606234, -
0.8225919629272787, 0.7475042084254954, -0.1799361158318473, -
1.5959201381576185, 0.6763030076403654, 0.7973787441173146, -
0.002822181103139224, -1.6809334736845545, -0.933374440045094,
0.05867855020213516, 1.2620758768187224, -0.6628377960393037,
1.3205514392352897, -0.3241539564812195, -1.252475305041749,
0.08871579653953558, 0.9118312236991867, -0.8399483947828849, -
0.6313431004803898, -0.6972985943133532, -1.453340048592718,
1.3580252253772858, -1.5824402455288722, -0.5815712900690646,
1.0866688562688684, -0.49086169609974084, 0.1353734497756709,
1.4897753206660764, -0.07035319597090053, 0.9262213835132865,
1.4455261318661783, 0.26941799128670463, 0.05467863036351897, -
1.4421935085607334, -1.4031936830528209, -0.7862831098220062, -
0.34590389050435605, -0.3915941553002187, 0.812965789177144, -
0.4623969472320197, -0.7866049584932192, 0.12064533670506865,
1.144348499847785, -0.14104718081647616, 0.8382583739153368, -
0.9633101517276529, 0.30143262258437387, 0.5966768658932305, -
0.9077152655980604, -0.5699203788512518, 0.6148129196077298,

```
1.5107160442848029, 1.5293592974835288, -1.6416693642947298,
0.3883462435340215, -0.9223086032022407, -1.4169358940469976, -
0.13716771740311293, 0.2765751056855065, 1.1554751138734813, -
1.2221021974601962, 0.13800379607319777, 1.641844310000692,
1.6251954394360584, 0.7511289694532917, -1.3185032140637494, -
1.3921421205181175, 0.3723588697422394, -0.010011607204937168, -
1.0553558746605722, 1.6771659532457746, 1.2531915227292725,
0.0145370651630044 1, -0.048869251060470666, 0.7365521137628274,
1.0619817951092039, -1.3915130833923677, -0.9031627627635117,
0.4606238449669017, -0.9172332350952933, 1.443557640807837,
1.6131568549891175, -1.592122821292571, 1.1222643958056406, -
0.9389081792360676, 0.5763486914582295, -0.6735822660057439,
0.2562998259355084 3, 0.6799439812642811, 1.5386543820168659,
1.3314455240708096, 1.4325260654376981, 0.054993852988579374]
```

Задание 5. Получить *LUP*-разложение матрицы A и решить полученную систему.

```
start_time = time.time()
L, U, P = GetLUP(A)
print("Время построения LUP: ", (time.time() - start_time))
start_time = time.time()
resultLUP = FindSolutionLUP(L, U, P, b)
print("Время решения: ", (time.time() - start_time))
print("Решение, полученное через LUP: ", resultLUP)
```

```
Время построения LUP:  1.8120169639587402
Время решения:  0.023640871047973633
Решение, полученное через LUP:  [0.6566520880736906, -
0.23015396915304304, 0.8719050790323561, 0.03889790878458685, -
0.0760423046955595, 0.7721991480879276, -0.9384746076998391,
1.1763115132406288, -1.350779611127787, -1.6569177393456578, -
0.8041701327377627, -0.8746041525873356, -1.094815859077489, -
1.574074642133234, -1.5111472965494799, -0.11258947259761329, -
0.5577354403269894, 0.17795955713729097, -1.4793268114463867, -
1.5093983615762903, -0.7033552671333629, 0.7469712701273156,
0.17860995804232313, -0.9479902679879054, 0.21053898437820628, -
1.6046163694620907, 0.09121873608015331, -0.2758297036696052, -
0.7300422645917516, -0.98051279957075, -0.6458456790615861, -
0.6076642701604529, 0.3342808176944129, 1.273672560012443,
1.0068775124101994, 0.004215003943413412, -1.5789447043023552,
1.035452268345909, -1.6733740788512388, -0.21883758615992094,
1.506393660911375, -0.9283957151128618, 0.7714893395480991,
0.6819383237977343, 1.369939960461185, -0.24655231785576084, -
0.5003655030263532, -1.463993521573813 1, -0.9078115059609083,
0.4004934506652631, -0.9598823226045747, 1.28856521855231,
1.6736365609380068, -1.239687373587186, 0.43874778062249054,
0.12476954356174651, 0.2950058115098234, 0.527943224400817,
0.6463619700492644, 0.5013881089948089, 0.27203464366399027, -
0.4794930863639285, -0.9420405208248467, -0.7622833817080795, -
0.5036210640529873, -1.2865881300685464, -1.2627275169627676, -
```

1.4901062128304619, -1.055544966774046, 0.9050305997444769,
1.1979169001504417, -1.0719694064816871, -0.43988910643900253, -
0.8271635839058396, 1.4964656565384729, 0.5630264307169506, -
1.6088472486336698, 1.186487039991519, -0.35053779024534754,
1.1909654190601466, 1.0794328689664996, -0.04532560017954723,
0.08350639470739304, -1.3251951746491877, 0.7029877673239672, -
0.9170620217096689, 1.3583690441678182, 0.8687998018701621, -
0.8076310098916317, -1.4581446472261512, 1.0455048471423471,
0.3522184766305131, 0.6172514400289991, -1.6500363040701749,
0.5199579627623379, -0.3592840823260098, -0.5021789039155587,
1.505749241799568, 0.9606489566335502, -0.5352524353466384, -
0.281358783722903, -1.1531343515794872, 1.2656099594416594,
0.6741477856125321, -0.8556905246580938, 0.9201219232495522,
0.2505555889794707, -1.1486326518674785, 0.08845349720963629,
0.4827894922992609, 1.1718895557632538, -0.8582132233710051,
0.8355045742968255, -0.001356508368608711, -0.8062125167679911, -
1.0109302810109, 0.7554063756923799, -0.2555203266346875,
1.5202231342493282, 1.5479529402765289, 0.30374914637222805, -
0.24880972641921845, -0.5318071662735091, 0.2624026093691663,
0.17891724585872207, 1.318873840542769, -1.6079924072429994,
1.5739463276075154, 1.4672874254194495, 0.09154033535685001, -
1.589984953511826, -0.15001902185305946, 0.6601207249715206,
1.5387219541727921, 1.275170035233411, -1.2872083927193863, -
1.4057165081812741, 0.8516600641288813, 1.2240979993347745,
0.436890433951596, 1.5841752475699915, -0.9885212448641064,
1.6080889200481934, -0.3908623951274674, 1.4287481702672802, -
1.2130417569955652, 0.27715922445716756, -0.6963307104299437, -
1.296612131141828, -0.5328686379866263, 1.1678567671302598,
0.6625521235273334, 0.762689201158575, -0.6770712985654163, -
1.1080400762661284, 0.46621565170007123, -0.6383378249033109,
1.1897609160322407, 0.6380274709985786, 0.21023759931197744, -
0.6625288294284891, 0.5140192138338581, 0.9856140306606226, -
0.8225919629272788, 0.7475042084254958, -0.17993611583184715, -
1.5959201381576162, 0.6763030076403644, 0.7973787441173134, -
0.0028221811031392792, -1.6809334736845536, -0.933374440045094,
0.05867855020213516, 1.2620758768187226, -0.6628377960393046,
1.320551439235289, -0.3241539564812195, -1.252475305041748,
0.08871579653953553, 0.911831223699187, -0.8399483947828846, -
0.6313431004803896, -0.6972985943133522, -1.453340048592719,
1.3580252253772862, -1.5824402455288729, -0.5815712900690639,
1.0866688562688678, -0.49086169609974084, 0.13537344977567092,
1.4897753206660778, -0.07035319597090056, 0.9262213835132862,
1.4455261318661765, 0.2694179912867048, 0.054678630363518964, -
1.4421935085607327, -1.403193683052823, -0.7862831098220064, -
0.3459038905043562, -0.39159415530021907, 0.8129657891771435, -
0.4623969472320195, -0.7866049584932198, 0.1206453367050687,
1.1443484998477857, -0.14104718081647602, 0.8382583739153359, -
0.9633101517276533, 0.3014326225843734, 0.5966768658932315, -
0.9077152655980603, -0.5699203788512515, 0.6148129196077297,
1.510716044284801, 1.5293592974835293, -1.6416693642947304,

```
0.38834624353402125, -0.9223086032022406, -1.4169358940469954, -
0.13716771740311295, 0.27657510568550664, 1.155475113873481, -
1.222102197460198, 0.13800379607319788, 1.6418443100006936,
1.6251954394360573, 0.751128969453291, -1.3185032140637494, -
1.392142120518118, 0.3723588697422392, -0.010011607204937144, -
1.0553558746605707, 1.6771659532457766, 1.2531915227292731,
0.014537065163004383, -0.04886925106047077, 0.7365521137628284,
1.061981795109204, -1.3915130833923692, -0.9031627627635123,
0.4606238449669016, -0.9172332350952935, 1.4435576408078374,
1.613156854989117, -1.5921228212925713, 1.1222643958056404, -
0.938908179236067, 0.5763486914582296, -0.6735822660057438,
0.2562998259355084, 0.6799439812642812, 1.5386543820168634,
1.3314455240708092, 1.4325260654376986, 0.054993852988579416]
```

Задание 6. Решить СЛАУ $Ax=b$ методом квадратного корня. Выписать LDL^T-разложение матрицы системы.

```
start_time = time.time()
L, D, LT = FindLDLT(A)
resultLDLT = FindSolutionLDLT(L, D, LT, b)
print("Время решения: ", (time.time() - start_time))
print("Решение, полученное чер LDLT: ", resultLDLT)
start_time = time.time()
resultSQRT = FindSolutionSQRT(A, b)
print("Время решения: ", (time.time() - start_time))
print("Решение, полученное методом квадратного корня: ", resultSQRT)
```

```
Время решения:  8.851962804794312
Решение, полученное чер LDLT:  [0.6566520880736906, -
0.23015396915304306, 0.8719050790323561, 0.03889790878458686, -
0.07604230469555952, 0.7721991480879276, -0.938474607699839,
1.1763115132406288, -1.3507796111277872, -1.6569177393456578, -
0.8041701327377626, -0.8746041525873356, -1.0948158590774892, -
1.5740746421332341, -1.5111472965494799, -0.11258947259761329, -
0.5577354403269895, 0.17795955713729097, -1.479326811446387, -
1.5093983615762903, -0.7033552671333629, 0.7469712701273155,
0.17860995804232305, -0.9479902679879054, 0.2105389843782062, -
1.6046163694620907, 0.09121873608015343, -0.2758297036696052, -
0.7300422645917517, -0.98051279957075, -0.6458456790615861, -
0.6076642701604529, 0.3342808176944129, 1.2736725600124432,
1.0068775124101996, 0.00421500394341346, -1.5789447043023552,
1.035452268345909, -1.6733740788512386, -0.2188375861599208,
1.506393660911375, -0.9283957151128618, 0.7714893395480991,
0.6819383237977343, 1.369939960461185, -0.24655231785576084, -
0.5003655030263533, -1.463993521573813 1, -0.9078115059609084,
0.4004934506652631, -0.9598823226045747, 1.2885652185523098,
1.6736365609380068, -1.239687373587186, 0.4387477806224905,
0.12476954356174645, 0.29500581150982347, 0.5279432244008169,
0.6463619700492645, 0.5013881089948089, 0.27203464366399027, -
0.4794930863639285, -0.9420405208248468, -0.7622833817080796, -
0.5036210640529875, -1.2865881300685464, -1.2627275169627679, -
```

1.490106212830462, -1.0555449667740457, 0.9050305997444769,
1.197916900150442, -1.0719694064816871, -0.4398891064390026, -
0.8271635839058394, 1.4964656565384729, 0.5630264307169506, -
1.6088472486336698, 1.186487039991519, -0.3505377902453476,
1.1909654190601469, 1.0794328689664994, -0.04532560017954729,
0.08350639470739302, -1.3251951746491877, 0.7029877673239673, -
0.9170620217096689, 1.3583690441678182, 0.8687998018701623, -
0.8076310098916317, -1.4581446472261514, 1.0455048471423471,
0.35221847663051314, 0.6172514400289991, -1.6500363040701747,
0.5199579627623377, -0.3592840823260097, -0.5021789039155587,
1.505749241799568, 0.9606489566335501, -0.5352524353466385, -
0.28135878372290307, -1.1531343515794872, 1.2656099594416594,
0.6741477856125321, -0.8556905246580938, 0.9201219232495522,
0.2505555889794708, -1.1486326518674785, 0.08845349720963626,
0.48278949229926094, 1.1718895557632538, -0.8582132233710051,
0.8355045742968256, -0.0013565083686086707, -0.8062125167679912, -
1.0109302810108998, 0.75540637569238, -0.2555203266346875,
1.5202231342493282, 1.5479529402765289, 0.30374914637222805, -
0.24880972641921847, -0.5318071662735091, 0.2624026093691663,
0.17891724585872207, 1.3188738405427687, -1.6079924072429994,
1.5739463276075154, 1.4672874254194495, 0.09154033535685, -
1.5899849535118258, -0.15001902185305943, 0.6601207249715205,
1.5387219541727923, 1.275170035233411, -1.287208392719386, -
1.405716508181274, 0.8516600641288813, 1.2240979993347745,
0.436890433951596, 1.5841752475699915, -0.9885212448641064,
1.6080889200481936, -0.3908623951274674, 1.4287481702672802, -
1.213041756995565, 0.2771592244571676, -0.6963307104299437, -
1.296612131141828, -0.5328686379866262, 1.1678567671302598,
0.6625521235273334, 0.762689201158575, -0.6770712985654163, -
1.1080400762661287, 0.46621565170007123, -0.6383378249033108,
1.1897609160322404, 0.6380274709985787, 0.21023759931197744, -
0.662528829428489, 0.5140192138338581, 0.9856140306606226, -
0.8225919629272788, 0.7475042084254958, -0.17993611583184718, -
1.595920138157616, 0.6763030076403646, 0.7973787441173135, -
0.002822181103139306, -1.6809334736845536, -0.933374440045094,
0.05867855020213514, 1.2620758768187226, -0.6628377960393046,
1.320551439235289, -0.3241539564812194, -1.252475305041748,
0.08871579653953553, 0.9118312236991871, -0.8399483947828846, -
0.6313431004803897, -0.6972985943133522, -1.4533400485927193,
1.3580252253772862, -1.5824402455288729, -0.5815712900690639,
1.0866688562688678, -0.4908616960997408, 0.13537344977567092,
1.4897753206660778, -0.07035319597090059, 0.9262213835132862,
1.4455261318661765, 0.2694179912867048, 0.05467863036351897, -
1.4421935085607325, -1.403193683052823 3, -0.786283109822006 4, -
0.3459038905043563, -0.39159415530021907, 0.8129657891771435, -
0.46239694723201946, -0.7866049584932198, 0.12064533670506873,
1.1443484998477857, -0.141047180816476, 0.8382583739153358, -
0.9633101517276533, 0.3014326225843734, 0.5966768658932314, -
0.9077152655980602, -0.5699203788512515, 0.6148129196077299,
1.5107160442848013, 1.5293592974835293, -1.6416693642947304,

0.3883462435340213, -0.9223086032022406, -1.4169358940469954, -
0.13716771740311295, 0.2765751056855067, 1.155475113873481, -
1.222102197460198, 0.1380037960731979, 1.6418443100006934,
1.6251954394360573, 0.751128969453291, -1.3185032140637496, -
1.3921421205181177, 0.37235886974223914, -0.010011607204937163, -
1.0553558746605707, 1.6771659532457766, 1.2531915227292731,
0.014537065163004355, -0.04886925106047078, 0.7365521137628284,
1.0619817951092043, -1.391513083392369, -0.9031627627635123,
0.4606238449669016, -0.9172332350952935, 1.4435576408078372,
1.613156854989117, -1.592122821292571, 1.1222643958056404, -
0.9389081792360668, 0.5763486914582298, -0.6735822660057439,
0.2562998259355084, 0.6799439812642812, 1.5386543820168632,
1.3314455240708092, 1.4325260654376986, 0.054993852988579416]
Время решения: 2.2101714611053467
Решение, полученное методом квадратного корня: [0.6566520880736905, -
0.2301539691530431, 0.8719050790323559, 0.038897908784586856, -
0.0760423046955596, 0.7721991480879276, -0.9384746076998391,
1.1763115132406288, -1.350779611127787, -1.6569177393456578, -
0.8041701327377625, -0.8746041525873357, -1.094815859077489, -
1.5740746421332343, -1.5111472965494805, -0.11258947259761327, -
0.5577354403269894, 0.17795955713729097, -1.4793268114463867, -
1.5093983615762903, -0.7033552671333627, 0.7469712701273155,
0.17860995804232316, -0.9479902679879054, 0.2105389843782062, -
1.6046163694620907, 0.09121873608015342, -0.2758297036696052, -
0.7300422645917517, -0.98051279957075, -0.645845679061586, -
0.6076642701604529, 0.33428081769441287, 1.2736725600124432,
1.0068775124101994, 0.004215003943413422, -1.5789447043023548,
1.035452268345909, -1.6733740788512388, -0.2188375861599208 8,
1.506393660911375, -0.9283957151128618, 0.7714893395480992,
0.6819383237977343, 1.369939960461185, -0.2465523178557609, -
0.5003655030263533, -1.4639935215738131, -0.9078115059609084,
0.40049345066526315, -0.9598823226045746, 1.2885652185523095,
1.6736365609380066, -1.2396873735871858, 0.4387477806224905,
0.12476954356174652, 0.2950058115098234, 0.527943224400817,
0.6463619700492644, 0.501388108994809, 0.2720346436639902, -
0.47949308636392857, -0.942040520824847, -0.7622833817080795, -
0.5036210640529873, -1.2865881300685464, -1.2627275169627679, -
1.490106212830462, -1.0555449667740457, 0.905030599744477,
1.1979169001504415, -1.0719694064816871, -0.4398891064390026, -
0.8271635839058393, 1.496465656538473, 0.5630264307169506, -
1.60884724863367, 1.186487039991519, -0.3505377902453475,
1.1909654190601469, 1.0794328689664994, -0.04532560017954726,
0.08350639470739298, -1.3251951746491877, 0.7029877673239673, -
0.9170620217096688, 1.3583690441678182, 0.8687998018701624, -
0.8076310098916317, -1.4581446472261517, 1.0455048471423476,
0.3522184766305131, 0.617251440028999, -1.6500363040701749,
0.5199579627623377, -0.3592840823260097, -0.5021789039155586,
1.505749241799568, 0.96064895663355, -0.5352524353466384, -
0.28135878372290296, -1.1531343515794872, 1.2656099594416594,
0.6741477856125321, -0.8556905246580937, 0.9201219232495521,

0.2505555889794707, -1.1486326518674788, 0.08845349720963626,
0.482789492299261, 1.1718895557632538, -0.8582132233710054,
0.8355045742968258, -0.001356508368608659, -0.8062125167679912, -
1.0109302810109, 0.7554063756923799, -0.2555203266346875,
1.5202231342493284, 1.547952940276529, 0.30374914637222816, -
0.2488097264192184, -0.5318071662735091, 0.2624026093691663,
0.17891724585872204, 1.3188738405427687, -1.6079924072429996,
1.5739463276075147, 1.4672874254194497, 0.09154033535685, -
1.589984953511826, -0.15001902185305951, 0.6601207249715206,
1.5387219541727926, 1.2751700352334108, -1.2872083927193863, -
1.4057165081812741, 0.8516600641288814, 1.2240979993347745,
0.43689043395159605, 1.5841752475699915, -0.9885212448641064,
1.6080889200481936, -0.39086239512746745, 1.4287481702672802, -
1.2130417569955647, 0.2771592244571676, -0.6963307104299438, -
1.2966121311418282, -0.5328686379866262, 1.16785676713026,
0.6625521235273333, 0.7626892011585752, -0.677071298565416, -
1.1080400762661287, 0.4662156517000713, -0.6383378249033107,
1.1897609160322409, 0.6380274709985787, 0.2102375993119774, -
0.662528829428489, 0.5140192138338581, 0.9856140306606226, -
0.8225919629272789, 0.7475042084254958, -0.17993611583184718, -
1.595920138157616, 0.6763030076403646, 0.7973787441173135, -
0.002822181103139323, -1.6809334736845531, -0.933374440045094,
0.05867855020213516, 1.2620758768187224, -0.6628377960393044,
1.320551439235289, -0.3241539564812194, -1.2524753050417479,
0.08871579653953551, 0.9118312236991871, -0.8399483947828845, -
0.6313431004803894, -0.6972985943133522, -1.4533400485927195,
1.3580252253772858, -1.5824402455288733, -0.5815712900690639,
1.0866688562688678, -0.4908616960997408, 0.13537344977567092,
1.4897753206660778, -0.07035319597090058, 0.9262213835132862,
1.4455261318661765, 0.26941799128670474, 0.05467863036351897, -
1.4421935085607327, -1.4031936830528233, -0.7862831098220064, -
0.34590389050435627, -0.391594155300219, 0.8129657891771433, -
0.4623969472320194, -0.78660495849322, 0.12064533670506872,
1.1443484998477857, -0.141047180816476, 0.8382583739153356, -
0.9633101517276536, 0.3014326225843734, 0.5966768658932314, -
0.9077152655980602, -0.5699203788512515, 0.6148129196077297,
1.5107160442848016, 1.5293592974835288, -1.6416693642947309,
0.38834624353402136, -0.9223086032022406, -1.4169358940469952, -
0.13716771740311298, 0.27657510568550664, 1.1554751138734811, -
1.222102197460198, 0.13800379607319788, 1.6418443100006934,
1.6251954394360573, 0.7511289694532911, -1.3185032140637494, -
1.3921421205181181, 0.3723588697422393, -0.010011607204937167, -
1.0553558746605707, 1.6771659532457763, 1.2531915227292727,
0.01453706516300437 6, -0.04886925106047078, 0.736552113762828 1,
1.0619817951092043, -1.391513083392369, -0.9031627627635123,
0.46062384496690156, -0.9172332350952934, 1.4435576408078372,
1.613156854989117, -1.592122821292571, 1.1222643958056404, -
0.9389081792360667, 0.5763486914582296, -0.673582266005744,
0.2562998259355084, 0.6799439812642811, 1.538654382016863,
1.3314455240708087, 1.4325260654376983, 0.0549938529885794]

Задание 7. Получить максимально точное решение СЛАУ $Ax=b$ методом релаксации (с параметром $1 - N/40$).

```
w = 1 - 3 / 40
eps = 0.00001
start_time = time.time()
solutionRelax = Relaxation(A, b, w, eps)
t = (time.time() - start_time)
print("Время решения: ", t)
print("Решение, полученное методом релаксации: ", solutionRelax[0])
```

```
Время решения:  0.23349380493164062
Решение, полученное методом релаксации:  [0.656651696654023, -
0.23015425602753298, 0.8719051517938513, 0.03889837567218211, -
0.07604257932962846, 0.7721992032215013, -0.9384747753417677,
1.1763117398670686, -1.3507799718124112, -1.6569172690492044, -
0.8041705410276329, -0.8746047041299091, -1.0948164178298305, -
1.574074288269166, -1.511146590917382, -0.1125893823821801, -
0.5577355365964684, 0.1779597773840834, -1.4793267927653648, -
1.5093982561811716, -0.7033556514168751, 0.746971042906901,
0.1786101124771228, -0.9479906724939196, 0.21053828236494854, -
1.6046164941568668, 0.09121886096426984, -0.27582955794665837, -
0.7300422001856779, -0.9805127720286461, -0.6458455576958045, -
0.6076642998599715, 0.33428094007449083, 1.273672230358165,
1.0068784003786202, 0.0042149406777504915, -1.5789449207091455,
1.035452524299068, -1.6733738519167602, -0.2188374555816137,
1.5063934907118508, -0.9283952397297719, 0.7714895960869981,
0.6819383945024171, 1.3699397462231775, -0.24655224720279317, -
0.5003654955407826, -1.4639935042276462, -0.907811409462771,
0.4004932073074901, -0.9598819732330565, 1.2885650328718394,
1.6736361478952066, -1.2396879642648886, 0.43874788255462455,
0.12476958910736223, 0.2950063051845265, 0.5279429659855683,
0.646362632198865, 0.5013883153830047, 0.27203447360849686, -
0.4794930166242165, -0.942040135255093, -0.7622830267111982, -
0.5036211076958634, -1.2865881834739068, -1.2627271794174735, -
1.4901059270943573, -1.0555449748258452, 0.9050303306599109,
1.1979165542341057, -1.0719693157082042, -0.43988880365984634, -
0.8271636512112099, 1.4964658899156045, 0.5630263110597873, -
1.6088470309110185, 1.186487168880523, -0.3505376989263195,
1.1909650579843776, 1.0794325068495525, -0.04532552443613944,
0.0835064804548612, -1.3251951358129324, 0.7029876751449217, -
0.9170620139309638, 1.3583689457850567, 0.8687998295714355, -
0.8076307401964468, -1.4581442118830863, 1.045504284002871,
0.3522182032862242, 0.6172514885643171, -1.650036280427908,
0.51995793988563, -0.35928401802510956, -0.5021788143752005,
1.5057490904769215, 0.9606491086975175, -0.5352522451777858, -
0.28135865667291005, -1.1531339647726904, 1.2656100759773927,
0.6741474463745855, -0.8556906187712259, 0.9201221079067085,
0.25055566311378974, -1.148633027040904, 0.08845351785664513,
0.4827896354827479, 1.171889570031582, -0.8582132051500811,
```

0.8355046340657801, -0.001356610336734187, -0.8062121751612905, -
1.0109301546082665, 0.7554067406839143, -0.255520424554948,
1.5202228301505125, 1.5479530269878052, 0.3037492182666433, -
0.24881013021075538, -0.5318072338020549, 0.2624026451400377,
0.17891731719925907, 1.3188735917546546, -1.6079920853604168,
1.5739465065886074, 1.4672867030702852, 0.09154028850832586, -
1.58998513215361, -0.15001922508789478, 0.6601209121197231,
1.5387221190268148, 1.2751701222628284, -1.2872085664889976, -
1.4057158790633315, 0.8516604186390148, 1.2240980922670737,
0.43689041463148376, 1.5841751911356696, -0.9885213122169669,
1.6080889489778694, -0.39086261231313446, 1.4287480258667868, -
1.2130418814079826, 0.2771592084777834, -0.6963308687216161, -
1.296612127165425, -0.5328684361474332, 1.167856778437221,
0.6625522124006853, 0.762689096063192, -0.6770712459032828, -
1.1080398864005028, 0.46621572873280914, -0.6383379072670475,
1.189760617357263, 0.6380274967155546, 0.21023745703409658, -
0.6625287628588454, 0.5140189804171514, 0.9856137823631267, -
0.8225918832742504, 0.7475044807783263, -0.17993598988241336, -
1.5959200783990277, 0.6763028843468484, 0.7973787345507882, -
0.0028222161088410085, -1.6809334581690567, -0.933374411329512,
0.05867855288601582, 1.26207566967265, -0.6628377457763943,
1.3205513463276224, -0.32415394309571666, -1.2524752463439708,
0.08871581658508632, 0.9118311758983272, -0.8399483597416426, -
0.6313431253401848, -0.6972986738232223, -1.4533399837872223,
1.3580251243893646, -1.5824403662487199, -0.5815710903839549,
1.086668592383063, -0.4908615766138245, 0.1353731611085961 2,
1.4897750774052685, -0.07035336009430575, 0.9262215342425981,
1.4455261555213343, 0.26941792583505547, 0.05467848311559899, -
1.4421934137354286, -1.403193445530155, -0.7862829896745426, -
0.3459037619458252, -0.3915942093246481, 0.8129658116982603, -
0.46239676004242836, -0.7866049231996665, 0.12064546593466972,
1.1443482873406599, -0.14104710028917075, 0.8382584625886229, -
0.9633103918415378, 0.30143267741005075, 0.5966769269799785, -
0.9077151818919346, -0.5699203055289752, 0.6148130115350203,
1.5107159111909492, 1.5293591751557531, -1.6416693789066816,
0.38834614923668836, -0.9223085493581725, -1.4169359331912679, -
0.1371677162495189, 0.27657515397836413, 1.1554750801797764, -
1.222102231644393, 0.13800378820291534, 1.6418440564848322,
1.6251954268824265, 0.7511288209227468, -1.3185031562343792, -
1.392142068788928, 0.3723589356498418, -0.01001149644914716, -
1.0553557404449616, 1.6771658893107215, 1.2531913991704686,
0.014537033338644425, -0.04886927690805759, 0.7365520574652569,
1.0619817621184162, -1.3915130292454876, -0.9031627546551281,
0.46062374057982836, -0.9172332289247446, 1.443557554779562,
1.6131567734371597, -1.592122773571011, 1.1222643036356914, -
0.9389080876021314, 0.5763487597051846, -0.6735823007051095,
0.25629982844993676, 0.6799439208824547, 1.5386543091169276,
1.3314454321771392, 1.4325259957038206, 0.05499382077006516]

Доказательство сходимости Обратим внимание на построение матрицы A. Из него видно, что матрица A – матрица со строгим диагональным преобладанием. Тогда для матрицы B кубическая норма будет меньше 1, что является достаточным условием сходимости.

Задание 8. Для его выполнения был запущен файл **lab1.py**. Среднее арифметическое число обусловленности: 4.635315728644947 Минимальное число обусловленности: 4.548548376877214 Максимальное число обусловленности: 4.917341940443021 Среднее время нахождения обратной матрицы: 2.993214685916901 Среднее время решения СЛАУ методом Гаусса: $ 0.7435239553345154$ Нормы разности решения методом Гаусса с точным решением: Средняя: 0.000000000000004 Минимальная:0.000000000000003 Максимальная: 0.000000000000006 Среднее время построения LUP-разложения: 0.869851973056793 Среднее время решения СЛАУ $Ax=b$: 0.010299108028412 Нормы разности решения с помощью LUP-разложения с точным решением: Средняя: 0.000000000000003 Минимальная: 0.000000000000002 Максимальная: 0.000000000000005 Среднее время решения СЛАУ методом квадратного корня: 4.214680202007294 Нормы разности решения методом квадратного корня с точным решением: Средняя: 0.000000000000003 Минимальная:0.000000000000002 Максимальная: 0.000000000000005 Среднее время решения СЛАУ методом релаксации: 0.107509365081787 Среднее количество итераций метода релаксации: 7.950000000000000 Минимальное количество итераций метода релаксации: 7.000000000000000 Максимальное количество итераций метода релаксации: 9.000000000000000 Нормы разности решения методом релаксации с точным решением: Средняя: 0.000000815610588 Минимальная:0.000000191769153 Максимальная: 0.000001307483497

Задание 9.

Заполним матрицу A_1.

```
A1 = [[V * V + 15, V - 1, -1, -2 ],
      [V - 1, -15 - V * V, -V + 4, -4],
      [-1, -V + 4, V * V + 8, -V],
      [-2, -4, -V, V * V + 10]]
N1 = 4
print("-------------------------A1-------------------------")
for i in range(N1):
  for j in range(N1):
    print('{:.3f}'.format(A1[i][j]), end = '\t\t')
  print()
```

```
-------------------------A1-------------------------
24.000          2.000      -1.000          -2.000
2.000      -24.000          1.000      -4.000
-1.000          1.000      17.000          -3.000
-2.000          -4.000          -3.000          19.000
```

Построим вектор x, который будет решением для новой системы. Умножим матрицу A_1 на вектор x и получим вектор b_1.

```
x1 =  [[0] * 1 for i in range(N1)]
for i in range(N1):
  x1[i][0] = random.uniform(Lborder, Rborder)
b1 = Multiple(A1, x1, int(N1), int(N1), int(1))
print("Вектор b1: ", b1)
print("Точное решение системы: ", x1)
```

```
Вектор b1:  [-20.6701693799814, -1.921360528977285,
25.732972938773916, -14.301400180234308]
Точное решение системы:  [[-0.8684606464198391],
[0.16332902754626377], [1.347667088989432], [-0.5969475839958229]]
```

Найдём число обусловленности матрицы A_1.

```
A1Invers = GetA_1(A1)
norm1 = GetNorm(A1)
norm1Invers = GetNorm(A1Invers)
EA1 = norm1 * norm1Invers
print("Число обусловленности:", EA1)
```

```
Число обусловленности: 2.3808128259276824
```

Решим СЛАУ $A_1 x = b_1$ методом Гаусса с выбором главного элемента по столбцу.

```
resultGaussA1 = FindSolutionGauss(A1, b1)
print("Решение, полученное методом Гаусса: ", resultGaussA1)
```

```
Решение, полученное методом Гаусса:  [-0.8684606464198392,
0.16332902754626377, 1.3476670889894324, -0.5969475839958229]
```

Получить LUP-разложение матрицы A_1 и решить полученную систему.

```
L, U, P = GetLUP(A1)
resultLUP1 = FindSolutionLUP(L, U, P, b1)
print("Решение, полученное через LUP: ", resultLUP1)
```

```
Решение, полученное через LUP:  [-0.8684606464198392,
0.16332902754626377, 1.347667088989432, -0.596947583995823]
```

Решить СЛАУ $A_1 x = b$ методом квадратного корня. Выписать LDL^T-разложение матрицы системы.

```
L, D, LT = FindLDLT(A1)
resultSQRT1 = FindSolutionSQRT(A1, b1)
print("Решение, полученное методом квадратного корня: ", resultSQRT1)
```

```
Решение, полученное методом квадратного корня:  [-0.8684606464198389,
0.1633290275462638, 1.3476670889894322, -0.596947583995823]
```

Получить максимально точное решение СЛАУ $A_1 x = b$ методом релаксации (с параметром $1 - N/40$).

```
w = 1 - 3 / 40
eps = 0.00001
solutionRelax1 = Relaxation(A1, b1, w, eps)
print("Решение, полученное методом релаксации: ", solutionRelax1[0])
```

```
Решение, полученное методом релаксации:  [0.8426958805311321,
1.1115843118766924, -0.12315448171227683, 1.0090060351687125]
```

Теперь все тоже самое для матрицы A_2.

```
N  = 8
A2_ = [[1,  1 + V, 2 + V, 3 + V, 4 + V, 5 + V, 6 + V, 7 + V],
       [100* V,  1000 *V,  10000 * V,  100000 * V,  - 1000* V,  -10000
* V, -100000 * V, 1],
       [V,  -1 + V, -2 + V, -3 + V, -4 + V, -5 + V, -6 + V,  -7 + V],
       [V - 1000, 10 * V - 1000,  100* V - 1000, 1000* V - 1000, 10000
* V - 1000,  -V, -V + 1, -V + 2],
       [-2 * V, 0, -1, -2, -3, -4, -5, -6],
       [V - 2019, -V + 2020, V - 2021, -V + 2022, V - 2023, -V + 2024,
V - 2025, -V + 2026],
       [2 * V - 2000, 4 * V - 2005, 8 * V - 2010, 16 * V - 2015, 32 *
V - 2020, 2019 * V, -2020 * V, 2021 * V],
       [1020 - 2 * V, -2924 + 896 * V, 1212 + 9808 * V, -2736 + 98918
* V, 1404 - 11068 * V, -1523 - 8078 * V, 2625 - 102119 * V, -1327 +
1924 * V]]
A2T_ = [[0] * N for i in range(N)]
for i in range(N):
  for j in range(N):
    A2T_[i][j] = A2_[j][i]
A2 = Multiple(A2T_, A2_, N, N, N)
print("--------------------------A2-------------------------")
for l in range(N):
  for j in range(N):
    print((A2[l][j]), end = '\t\t')
  print()
print("-----------------------------------------------------------")
```

```
--------------------------A2-------------------------
10152543.0      1535566.0       48791190.0      385992164.0
     -54149402.0          -51266575.0          -381822232.0
     -11659429.0
1535566.0       18036954.0      83336718.0      836664330.0
     -29864982.0          -91913654.0          -820318072.0
     -9048186.0
48791190.0      83336718.0      1847071043.0          18005967600.0
          -1076627339.0         -1705196890.0         -18289016549.0
          120084240.0
385992164.0          836664330.0          18005967600.0
```

```
     176458529814.0        -10192066224.0        -16580861290.0
     -179294841510.0       1299366598.0
-54149402.0           -29864982.0           -1076627339.0         -
10192066224.0         1869022235.0          893249582.0
     10574363561.0         -157134580.0
-51266575.0           -91913654.0           -1705196890.0         -
16580861290.0         893249582.0           1604194832.0
     16782433346.0         -73707676.0
-381822232.0          -820318072.0          -18289016549.0        -
179294841510.0        10574363561.0         16782433346.0
     182293940027.0        -1391220892.0
-11659429.0           -9048186.0      120084240.0
     1299366598.0          -157134580.0          -73707676.0
     -1391220892.0         60610677.0
-------------------------------------------------------
```

Построим вектор x, который будет решением для новой системы. Умножим матрицу A_2 на вектор x и получим вектор b_2.

```
N = 8
x2 =  [[0] * 1 for i in range(N)]
for i in range(N):
  x2[i][0] = random.uniform(Lborder, Rborder)
b2 = Multiple(A2, x2, int(N), int(N), int(1))
print("Вектор b2: ", b2)
print("Точное решение системы: ", x2)
```

```
Вектор b2:  [-906182899.820053, -1969804169.5974312, -
43335457692.22401, -424845309062.7393, 25407909644.746643,
39761949300.92785, 431965570920.878, -3302211202.3498]
Точное решение системы:  [[1.0954509428231545], [-0.5472031131854598],
[1.2161214888350285], [-0.9536196298944906], [0.48694824352059585], [-
0.8318294401519049], [1.6083297968134793], [0.8483754695421919]]
```

Найдём число обусловленности матрицы A_2.

```
A2Invers = GetA_1(A2)
norm2 = GetNorm(A2)
norm2Invers = GetNorm(A2Invers)
EA2 = norm2 * norm2Invers
print("Число обусловленности:", EA2)
```

```
Число обусловленности: 1.6475233805374614e+16
```

Решить СЛАУ $A_2 x = b$ методом Гаусса с выбором главного элемента по столбцу.

```
resultGaussA2 = FindSolutionGauss(A2, b2)
print("Решение, полученное методом Гаусса: ", resultGaussA2)
```

```
Решение, полученное методом Гаусса:  [1.0951259753788973, -
0.3737317844455979, 1.2826295338117637, -1.0855167426842556,
```

```
0.5034057850328221, -1.240098136695826, 1.5254813915917937,
1.2145655133238786]
```

Получить LUP-разложение матрицы A_2 и решить полученную систему.

```
L, U, P = GetLUP(A2)
resultLUP2 = FindSolutionLUP(L, U, P, b2)
print("Решение, полученное через LUP: ", resultLUP2)
```

```
Решение, полученное через LUP:  [1.0952157659789408, -
0.4216043842616339, 1.2642752815876472, -1.0491173239369735,
0.4988640254833955, -1.1274290516310899, 1.5483449211761147,
1.1135087837056146]
```

Решить СЛАУ $A_2 x = b$ методом квадратного корня. Выписать $L D L^T$ - разложение матрицы системы.

```
L, D, LT = FindLDLT(A2)
resultLDLT2 = FindSolutionSQRT(A2, b2)
print("Решение, полученное методом квадратного корня: ", resultLDLT2)
```

```
Решение, полученное методом квадратного корня:  [1.0951177450652863, -
0.3694476705917497, 1.2842722566564337, -1.0887741195539198,
0.503812224593012, -1.2501806671576805, 1.523435338706729,
1.2236088602023822]
```

Получить максимально точное решение СЛАУ $A_2 x = b$ методом релаксации (с параметром $1 - N/40$).

```
w = 1 - 3 / 40
eps = 0.00001
solutionRelax2 = Relaxation(A2, b2, w, eps)
print("Решение, полученное методом релаксации: ", solutionRelax2[0])
```

```
Решение, полученное методом релаксации:  [-6.367692100752958, -
16.59835163050726, -20.089466556513138, -0.13916157282907066, -
0.8783418414790211, 0.7126186633234003, -0.016677328634546802, -
17.193074979273238]
```

Можно заметить, что для матрицы A_2 метод релаксации не работает, т.к. не все собственные значения матрицы B будут по модулю меньше единицы.

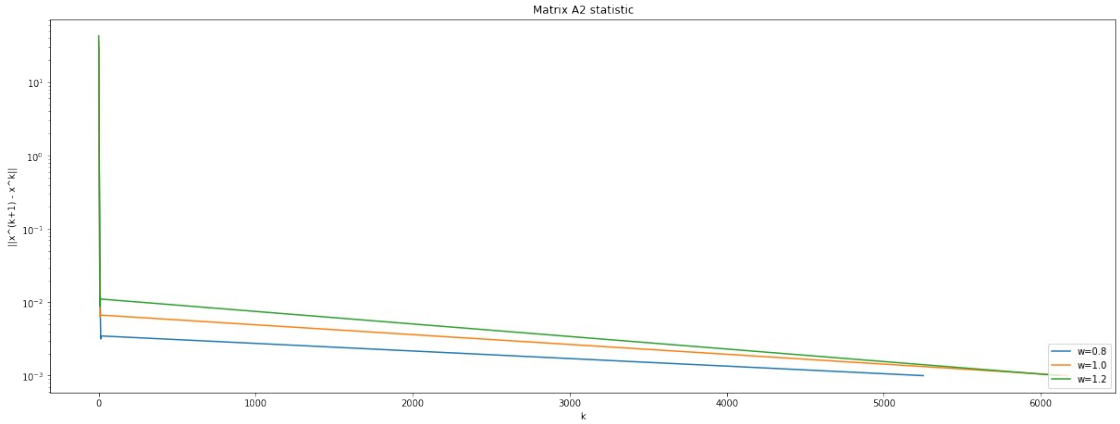
Задание 10

10.1 Решив несколько СЛАУ методом Гаусса с различным возмущением вектора b, получили следующие результаты: **Относительная погрешность 1:** 0.0016159749513193179 **Относительная погрешность 2:** 0.0002300492058692748 **Относительная погрешность 3:** 0.00000117946448184634 4 **Относительная погрешность 4:** 0.028318868459091543 Как видим, незначительное изменение вектора b

привело к довольно значимым погрешностям. *P.S. Изменнения вектора b можно посмотреть в* ***diagnostic.py****.*

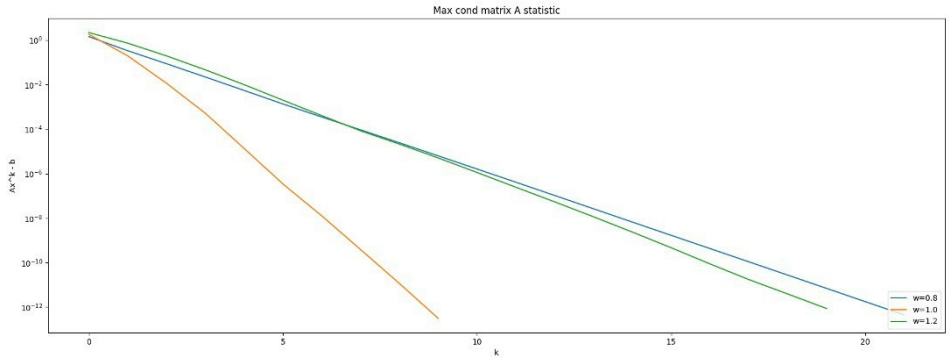
```
fig, (ax1) = plt.subplots(1)
eps = 0.001
ax1.plot(Relaxation(A2, b2, 0.8, eps)[2], label='w=0.8')
ax1.plot(Relaxation(A2, b2, 1.0, eps)[2], label='w=1.0')
ax1.plot(Relaxation(A2, b2, 1.2, eps)[2], label='w=1.2')
ax1.legend(loc='lower right')

ax1.set_yscale('log')
ax1.set_xlabel('k')
ax1.set_ylabel('||x^(k+1) - x^k||')
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9,
wspace=0.4, hspace=0.4)
ax1.title.set_text('Matrix A2 statistic')
fig = plt.gcf()
fig.set_size_inches(20, 7)
plt.show()
```



Можно видеть, что пробделано более 5000 операций, но ответ все равно получен с очень большой погрешностью.

Аналогичные действия проделаны с сохраненной матрицей.



Можно сделать вывод, что для сохраненной матрицы оптимальное значение равно 0.8.

```
import random
import math
import time
import matplotlib.pyplot as plt
```