```
#Exercise 6.2
#15.2.2
#author : chuanlu

import numpy as np
import pdb

#exercise6.2.2
def jacobi(A, b, x, tol):
    d = np.diag(A)
    d = [x ** -1 for x in d]
    D = np.diag(d)
    L = np.tril(A, -1)
    U = np.triu(A, 1)
    B = np.dot(D, L + U)
    g = np.dot(D, b)
    count = 0
    while True:
        new_x = np.dot(B, x) + g
        count += 1
        c = np.linalg.norm((new_x - x), np.inf)
        x = new_x
        if c < tol:
            print(count)
            print(c)
            return new_x

def sor(A, b, x, omega, tol):
    d = np.diag(A)
    D = np.diag(d)
    L = np.tril(A, -1)
    U = np.triu(A, 1)
    T = np.linalg.inv(D - omega * L)
    B = np.dot(T, (1 - omega) * D + omega * U)
    g = omega * T
    count = 0
    while True:
        new_x = np.dot(B, x) + g
        count += 1
        c = np.linalg.norm((new_x - x), np.inf)
        x = new_x
        if c < tol:
            print(omega)
            print(count)
            print(c)
            return new_x

def cg(A, b, x, tol):
    r = b - np.dot(A, x)
    count = 0
    x = [x]
    r = [r]
    while True:
        count += 1
        if count == 1:
            p = r[0]
        else:
            beta = np.dot(np.transpose(r[-1]), r[-1]) / np.dot(np.transpose(r[-2]), r[-2])
            p = r[-1] + beta * p
        alpha = np.dot(np.transpose(r[-1]), r[-1]) / np.dot(np.dot(np.transpose(p), A), p)
        x_new = x + alpha * p
        r.append(r[-1] - alpha * np.dot(A, p))
        c = np.linalg.norm((x_new - x), np.inf)
        x = x_new
        if c < tol:
            print(count)
            print(c)
            return x_new


def main():
    N = 10
    h = 1/(N+1)
    f = lambda x, y: (x ** 2 + y ** 2) * np.exp(x * y)

    #exercise 6.2.1
    x = np.linspace(0, 1, N + 2)
    y = np.linspace(0, 1, N + 2)

    rank = N * N
    A = np.diag([4 for i in range(rank)]) + \
        np.diag([-1 for i in range(rank - 1)], -1) + \
        np.diag([-1 for i in range(rank - 1)], 1) + \
        np.diag([-1 for i in range(rank - N)], -N) + \
        np.diag([-1 for i in range(rank - N)], N)
    b = np.zeros(N * N)
    b = np.transpose(b)
    for i in range(N):
        for j in range(N):
            t = j + i * N
            b[t] = f(x[j + 1], y[i + 1]) * (h**2)
            if i == 0:
                b[t] += 1
            elif i == N - 1:
                b[t] += np.exp(x[j + 1])
            else:
                continue
            if j == 0:
                b[t] += 1
            elif j == N - 1:
                b[t] += np.exp(y[i + 1])
            else:
                continue
    print('exercise6.2.1')
    print(A)
    print(b)
    print('exercise6.2.2')
    print('jacobi')
    tol = 1e-5
    init = np.zeros(N*N).transpose() + 1
    result = jacobi(A, b, init, tol)
    print(result)
    print('SOR')
    omega_list = [1, 1.25, 1.5, 1.75]
    for omega in omega_list:
        result = sor(A, b, init, omega, tol)
        print(result)
    print('exercise6.2.3')
    print('CG')
    result = cg(A, b, init, tol)
    print(result)

if __name__ == '__main__':
    main()、
```

运行结果如下：

```
exercise6.2.1
[[ 4 -1  0 ...,  0  0  0]
 [-1  4 -1 ...,  0  0  0]
 [ 0 -1  4 ...,  0  0  0]
 ...,
 [ 0  0  0 ...,  4 -1  0]
 [ 0  0  0 ..., -1  4 -1]
 [ 0  0  0 ...,  0 -1  4]]
[  2.00013774e+00   1.00034720e+00   1.00070016e+00   1.00120015e+00
   1.00185075e+00   1.00265562e+00   1.00361846e+00   1.00474303e+00
   1.00603318e+00   2.10266222e+00   3.47198376e-04   5.64775812e-04
   9.33056355e-04   1.45939529e-03   2.15139108e-03   3.01689210e-03
   4.06400354e-03   5.30109444e-03   6.73680499e-03   8.38005387e-03
   7.00159347e-04   9.33056355e-04   1.32435574e-03   1.88555756e-03
   2.62873214e-03   3.56654382e-03   4.71227552e-03   6.07985433e-03
   7.68387790e-03   9.53964196e-03   1.20014860e-03   1.45939529e-03
   1.88555756e-03   2.49463268e-03   3.30367508e-03   4.33085514e-03
   5.59552078e-03   7.11826227e-03   8.92098019e-03   1.10269568e-02
   1.85075384e-03   2.15139108e-03   2.62873214e-03   3.30367508e-03
   4.19884196e-03   5.33869871e-03   6.74968259e-03   8.46033753e-03
   1.05014580e-02   1.29062419e-02   2.65562193e-03   3.01689210e-03
   3.56654382e-03   4.33085514e-03   5.33869871e-03   6.62175849e-03
   8.21476284e-03   1.01557352e-02   1.24862634e-02   1.52517895e-02
   3.61845968e-03   4.06400354e-03   4.71227552e-03   5.59552078e-03
   6.74968259e-03   8.21476284e-03   1.00352144e-02   1.22603677e-02
   1.49448954e-02   1.81493163e-02   4.74303469e-03   5.30109444e-03
   6.07985433e-03   7.11826227e-03   8.46033753e-03   1.01557352e-02
   1.22603677e-02   1.48370892e-02   1.79564477e-02   2.16975129e-02
   6.03317617e-03   6.73680499e-03   7.68387790e-03   8.92098019e-03
   1.05014580e-02   1.24862634e-02   1.49448954e-02   1.79564477e-02
   2.16107745e-02   2.60097865e-02   2.10266222e+00   1.20777616e+00
   1.32308160e+00   1.44957797e+00   1.58836335e+00   1.74064426e+00
   1.90774643e+00   2.09112652e+00   2.29238519e+00   4.99534639e+00]
exercise6.2.2
jacobi
587
9.89753129249e-06
[  6.19844679e-01  -2.18419336e-02   2.63443918e-01   1.26359626e-01
   1.91628934e-01   1.72204549e-01   1.45857628e-01   2.48835853e-01
  -2.74786408e-02   7.27640645e-01  -4.57405426e-01   2.04417522e-01
  -1.57603117e-01   4.06783027e-02  -6.32399975e-02  -2.36602269e-02
  -8.63781667e-04  -1.08991003e-01   1.39459066e-01  -3.23029609e-01
   2.78049886e-01  -1.80273059e-01   1.22786268e-01  -6.67906547e-02
   4.64435041e-02  -1.04642367e-02  -5.70890061e-03   5.38118957e-02
  -9.16229353e-02   1.55325104e-01  -1.50816696e-01   1.16744755e-01
  -8.51818432e-02   5.91113664e-02  -4.26796774e-02   2.83194230e-02
  -1.49660479e-02  -2.87533235e-03   2.55485832e-02  -4.63232476e-02
   5.43145177e-02  -4.92816327e-02   4.39361628e-02  -3.93336185e-02
   4.01127128e-02  -4.08725507e-02   4.56886167e-02  -4.58109650e-02
   4.75117672e-02  -3.89050186e-02   3.09770410e-02  -1.57552045e-02
   6.43072588e-04   1.74394887e-02  -3.34047894e-02   5.46694141e-02
  -7.43939802e-02   1.01340419e-01  -1.20416883e-01   1.36322125e-01
  -1.20945468e-01   8.36605568e-02  -4.46649210e-02   6.62947596e-03
   2.66975059e-02  -6.34232047e-02   1.04053816e-01  -1.54622214e-01
   2.08941794e-01  -2.49806559e-01   2.36403847e-01  -1.49249026e-01
   9.24027286e-02  -2.04303623e-02  -9.87749888e-03   7.64514673e-02
  -1.13775610e-01   2.16378818e-01  -2.96010228e-01   4.35675116e-01
  -4.20903015e-01   1.89799096e-01  -1.49217194e-01  -3.45042823e-04
  -3.47779690e-02  -1.08603102e-01   7.04499729e-02  -2.86298088e-01
   3.40974773e-01  -7.54308554e-01   8.27743387e-01  -3.31130747e-02
   3.22673796e-01   2.14705043e-01   2.68417711e-01   3.34754335e-01
   2.41801476e-01   5.35325722e-01  -5.68966823e-03   1.43883456e+00]
SOR
1
129
9.39455328365e-06
[[  3.09862101e-01  -1.10874412e-01   4.58707574e-02 ...,  -3.71368626e-04
    8.00734743e-04  -8.55954460e-04]
 [ -1.10874431e-01   3.49532678e-01  -1.27285595e-01 ...,  -1.57876921e-04
   -5.31173749e-04   8.00733881e-04]
 [  4.58708275e-02  -1.27285674e-01   3.56321121e-01 ...,   1.01442613e-03
   -1.57883535e-04  -3.71363701e-04]
 ...,
 [ -3.71170627e-04  -1.58171801e-04   1.01473381e-03 ...,   3.56321131e-01
   -1.27285606e-01   4.58707640e-02]
 [  8.00571322e-04  -5.30921835e-04  -1.58163439e-04 ...,  -1.27285663e-01
    3.49532671e-01  -1.10874408e-01]
 [ -8.55849644e-04   8.00566037e-04  -3.71165203e-04 ...,   4.58708412e-02
   -1.10874442e-01   3.09862108e-01]]
1.25
82
9.38581286303e-06
[[  3.09862111e-01  -1.10874481e-01   4.58708237e-02 ...,  -3.71287778e-04
    8.00679602e-04  -8.55942653e-04]
 [ -1.10874433e-01   3.49532753e-01  -1.27285675e-01 ...,  -1.58027259e-04
   -5.31059821e-04   8.00687755e-04]
 [  4.58708166e-02  -1.27285727e-01   3.56321190e-01 ...,   1.01463145e-03
   -1.58049571e-04  -3.71278722e-04]
 ...,
 [ -3.71206910e-04  -1.58098888e-04   1.01468371e-03 ...,   3.56321218e-01
   -1.27285698e-01   4.58708433e-02]
 [  8.00602059e-04  -5.30991534e-04  -1.58109930e-04 ...,  -1.27285708e-01
    3.49532728e-01  -1.10874464e-01]
 [ -8.55869435e-04   8.00616078e-04  -3.71206962e-04 ...,   4.58708515e-02
```

```
   -1.10874463e-01   3.09862136e-01]]
1.5
54
9.01359846291e-06
[[  3.09862132e-01  -1.10874369e-01   4.58708090e-02 ...,  -3.71146486e-04
    8.00556882e-04  -8.55810182e-04]
 [ -1.10874463e-01   3.49532600e-01  -1.27285644e-01 ...,  -1.58157280e-04
   -5.30941209e-04   8.00537631e-04]
 [  4.58708473e-02  -1.27285567e-01   3.56321147e-01 ...,   1.01469131e-03
   -1.58113969e-04  -3.71159223e-04]
 ...,
 [ -3.71215423e-04  -1.58114477e-04   1.01466338e-03 ...,   3.56321095e-01
   -1.27285602e-01   4.58707736e-02]
 [  8.00610365e-04  -5.30965503e-04  -1.58097472e-04 ...,  -1.27285602e-01
    3.49532642e-01  -1.10874396e-01]
 [ -8.55875438e-04   8.00591089e-04  -3.71211828e-04 ...,   4.58707816e-02
   -1.10874406e-01   3.09862087e-01]]
1.75
90
7.95773223646e-06
[[  3.09862087e-01  -1.10874428e-01   4.58707890e-02 ...,  -3.71214972e-04
    8.00597224e-04  -8.55869846e-04]
 [ -1.10874406e-01   3.49532674e-01  -1.27285623e-01 ...,  -1.58095916e-04
   -5.30976133e-04   8.00603351e-04]
 [  4.58707846e-02  -1.27285642e-01   3.56321123e-01 ...,   1.01466110e-03
   -1.58106202e-04  -3.71211078e-04]
 ...,
 [ -3.71236368e-04  -1.58115019e-04   1.01464583e-03 ...,   3.56321126e-01
   -1.27285642e-01   4.58707978e-02]
 [  8.00579086e-04  -5.30994149e-04  -1.58117985e-04 ...,  -1.27285642e-01
    3.49532658e-01  -1.10874422e-01]
 [ -8.55885388e-04   8.00593631e-04  -3.71222146e-04 ...,   4.58707943e-02
   -1.10874425e-01   3.09862100e-01]]
exercise6.2.3
CG
35
5.56182387346e-06
[[ 1.11261174  1.1236006   1.12003575  1.11825228  1.12096855  1.12931851
   1.14657453  1.18327922  1.27367188  1.53687082  1.32670848  1.26140779
   1.23758981  1.23080479  1.23445266  1.24707546  1.27008184  1.3081276
   1.36850414  1.44444053  1.39559631  1.35716723  1.337178    1.33146508
   1.33681057  1.35143192  1.37448591  1.40534416  1.44103984  1.4684107
   1.45336859  1.43355375  1.42116537  1.41918165  1.4272639   1.4437892
   1.46637361  1.49164338  1.51421642  1.52525419  1.51471331  1.50105435
```

```
    1.49286259   1.4943377    1.50597051   1.52575672   1.54998034   1.57352125
    1.5900075    1.59264921   1.57732536   1.56093612   1.55226422   1.55603235
    1.57232493   1.59794796   1.62752033   1.65399348   1.66914163   1.66510367
    1.63834701   1.61008361   1.59565931   1.6008717    1.6240102    1.65956829
    1.6999446    1.73563504   1.75497571   1.74502485   1.69725699   1.641328
    1.61470533   1.62218951   1.65652625   1.70815571   1.76701967   1.82136601
    1.85515648   1.84461366   1.7595849    1.6379648    1.59356458   1.60953621
    1.66328927   1.73935285   1.82635214   1.91281586   1.98171408   1.99699081
    1.85247071   1.55064494   1.50436789   1.55018056   1.6372403    1.74712814
    1.8712754    2.00387504   2.14028228   2.28315489]]
[Finished in 0.3s]
```

结果分析：

Jacobi 迭代的收敛速度最慢，其次为 C—S 迭代，再为 omega = 1.75 的 SOR 迭代，再为 omega = 1.25 的 SOR 迭代，再为 omega = 1.5 的 SOR 迭代，CG 方法的收敛速度最快。

此外，由 SOR 迭代的结果可以得知，最佳松弛因子位于 1.25 和 1.75 之间。