```python
import os, sys
sys.path.append(os.path.join(os.path.dirname(__file__), '../ch02'))
sys.path.append(os.path.join(os.path.dirname(__file__), '../ch03/program3_3'))

from math import sqrt

from program2_1 import Dvector
from program2_2 import Dmatrix
from program2_3 import input_matrix
from program2_4 import inner_product
from program3_3 import lu_decomp, lu_solve
from program9_2 import householder
from program9_3 import qr

N = 4

def main():
    eps = 10.0 ** -8.0

    a  = Dmatrix(1, N, 1, N) # 行列領域の確保

    with open("input_eigen.dat", "r") as fin:
        with open("result_eigen.dat", "w") as fout:
            input_matrix( a, 'A', fin, fout ) # 行列Aの入出力

            a_hh = householder(a, N) # ハウスホルダー法

            a_qr = qr( a_hh , eps, N) # QR 法
            print("固有値は")
            for i in range(1, N+1):
                print("{:10.7f}".format(a_qr[i][i]), end="\t")
            print()

            a_ii = inverse_iteration( a, a_qr, eps ) # 逆反復法
            print("固有ベクトルは")
            for i in range(1, N+1):
                print("[", end="")
                for j in range(1, N+1):
                    print("{:10.7f}".format(a_ii[j][i]), end="\t")
                print("]")


# 逆反復法
def inverse_iteration( a: Dmatrix, a_qr: Dmatrix, eps: float ) -> Dmatrix:
    mu = 0.0

    y    = Dvector(1, N)
    a_ii = a_qr.copy()

    for i in range(1, N+1):
        lambda_ = a_ii[i][i] # 近似固有値の代入
        y[i] = 1.0 # 初期値設定

        # 行列の作成およびLU分解
        lu = a.copy()
        for k in range(1, N+1):
            lu[k][k] -= lambda_
        lu, p = lu_decomp(lu, N) # LU分解

        # 逆反復法
        while True:
            muo = mu
            v = y.copy()
            v = lu_solve(lu, v, p, N) # 固有ベクトルの計算
            mu = inner_product(v, y) # 補正
            v2s = sqrt(inner_product(v, v))
            for j in range(1, N+1):
                y[j] = v[j] / v2s

            if abs((mu-muo)/mu) < eps:
                break

        # 結果の代入 (固有ベクトルはaのi列に)
        for j in range(1, N+1):
            a_ii[j][i] = y[j]

    return a_ii


if __name__ == "__main__":
    main()
```