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

from math import sqrt

from program2_1 import Dvector
from program2_2 import Dmatrix
from program2_3 import input_matrix

N = 4

def main():
    a = Dmatrix(1, N, 1, N)

    # ファイルのオープン
    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

            # 結果の出力
            print("Hessenberg 行列は")
            for i in range(1, N+1):
                for j in range(1, N+1):
                    print("{:10.7f}\t".format(a_hh[i][j]), end="")
                print()


# ハウスホルダー法
def householder(a: Dmatrix, n: int) -> Dmatrix:
    u = Dvector(1, n)
    f = Dvector(1, n)
    g = Dvector(1, n)

    a_hh = a.copy()

    for k in range(1, n-1):
        # v の計算
        for i in range(1, k+1):
            u[i] = 0.0
        for i in range(k+1, n+1):
            u[i] = a_hh[i][k]

        # s の計算
        ss = 0.0
        for i in range(k+2, n+1):
            ss += u[i]*u[i]
        if abs(ss) <= 0.0:       # 消去が必要ない場合の処理
            continue
        s = sqrt( ss + u[k+1]*u[k+1] )
        if u[k+1] > 0.0:
            s = -s

        # uの計算
        u[k+1] -= s
        uu = sqrt( ss + u[k+1]*u[k+1] )
        for i in range(k+1, n+1):
            u[i] /= uu

        # f, gの計算
        for i in range(1, n+1):
            f[i], g[i] = 0.0, 0.0
            for j in range(k+1, n+1):
                f[i] += a_hh[i][j]*u[j]
                g[i] += a_hh[j][i]*u[j]

        # gammaの計算
        gamma = 0.0
        for j in range(1, n+1):
            gamma += u[j]*g[j]

        # f, gの計算
        for i in range(1, n+1):
            f[i] -= gamma * u[i]
            g[i] -= gamma * u[i]

        # A の計算
        for i in range(1, n+1):
            for j in range(1, n+1):
                a_hh[i][j] -= 2.0*u[i]*g[j] + 2.0*f[i]*u[j]

    return a_hh


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