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

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

N = 4 # N次正方行列

def main():
    global N

    a = Dmatrix(1, N, 1, N) # 行列 a[1...N][1...N]
    b = Dvector(1, N) # b[1...N]

    # ファイルのオープン
    with open("input_cho.dat", "r") as fin:
        with open("output_cho.dat", "w") as fout:
            input_matrix( a, 'A', fin, fout ) # 行列 A の入出力
            input_vector( b, 'b', fin, fout ) # ベクトル b の入出力
            a_cd = cholesky_decomp( a )          # 修正コレスキー分解
            b_cs = cholesky_solve( a_cd, b )        # 前進代入・後退代入

            # 結果の出力
            fout.write("Ax=bの解は次の通りです\n")
            for i in range(1, N+1):
                fout.write("{:.6f}\t\n".format(b_cs[i]))


# 修正コレスキー分解
def cholesky_decomp(a: Dmatrix, N:int=N):
    a_cd = a.copy()
    for i in range(2, N+1):
        for j in range(1,i):
            a_cd[i][j] = (a_cd[i][j] - sum(( a_cd[i][k] * a_cd[k][k] * a_cd[j][k] for k in range(1, j) ))) / a_cd[j][j]
        a_cd[i][i] -= sum((a_cd[i][k] * a_cd[i][k] * a_cd[k][k] for k in range(1, j+1) ))
    return a_cd


# 修正コレスキー分解を利用して連立一次方程式を解く
def cholesky_solve(a_cd: Dmatrix, b: Dvector, N:int=N):
    b_cs = b.copy()
    # LDy = b
    b_cs[1] = b[1] / a_cd[1][1]
    for i in range(2, N+1):
        b_cs[i] = ( b_cs[i] - sum( ( a_cd[j][j] * a_cd[i][j] * b_cs[j] for j in range(1,i) ) ) ) / a_cd[i][i]

    # L^t x = y
    for i in range(N-1, 0, -1):
        b_cs[i] -= sum( ( a_cd[j][i] * b_cs[j] for j in range(i+1,N+1) ) )

    return b_cs


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