```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
from program2_8 import vector_norm_max

N    = 10           # N元方程式
EPS  = 10.0 ** -8.0 # epsilon の設定
KMAX = 100          # 最大反復回数

def main():
    omega = 1.22

    a  = Dmatrix(1, N, 1, N) # 行列 a[1...N][1...N]
    b  = Dvector(1, N)       # b[1...N]
    x0 = Dvector(1, N)       # x0[1...N]

    # ファイルのオープン
    with open("input_sp.dat", "r") as fin:
        with open("output_sp.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
            input_vector( x0, 'x0', fin, fout ) # 初期ベクトル x0 の入出力
            x = sor( a, b, x0, omega )          # SOR法

            # 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}\n".format(x[i]))


# SOR法
def sor(a: Dmatrix, b: Dvector, x0: Dvector, omega: float, N:int=N):
    k = 0

    x  = x0.copy()
    xo = Dvector(1, N) # xo[1...N]

    while True:
        # xo <- x_k, x <- x_{k+1}
        for i in range(1, N+1):
            xo[i] = x[i]        # x_k に x_(k+1) を代入

        # i=1 の処理
        x[1] = ( b[1] - sum( ( a[1][j] * xo[j] for j in range(2, N+1) ) ) ) / a[1][1]

        # i=2,3,...N の処理
        for i in range(2, N+1):
            s = sum( ( a[i][j] * x[j]  for j in range( 1 ,  i ) ) ) # i-1列までの和
            t = sum( ( a[i][j] * xo[j] for j in range(i+1, N+1) ) ) # i+1列以降の和
            x[i] = ( b[i] - s - t ) / a[i][i]
        # ここまではガウス・ザイデル法と同じ

        # SOR法
        for i in range(1, N+1):
            x[i] = xo[i] + omega * ( x[i] - xo[i] ) # 補正

        for i in range(1, N+1):
            xo[i] = xo[i] - x[i]
        eps = vector_norm_max(xo)
        k += 1

        if eps <= EPS or k >= KMAX:
            break

    if k == KMAX:
        print("答えが見つかりませんでした")
        exit(1)
    else:
        print(f"反復回数は{k}回です")       # 反復回数を画面に表示
        return x


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