```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

M = 6 # データのペア数
N = 3 # N次式で近似

def main():
    global M, N

    x = Dvector(1, M) # x[1...M]
    y = Dvector(1, M) # y[1...M]

    # ファイルのオープン
    with open("input_func.dat", "r") as fin:
        with open("output_func.dat", "w") as fout:
            input_vector( x, 'x', fin, fout ) # ベクトル x の入出力
            input_vector( y, 'y', fin, fout ) # ベクトル y の入出力

            least_square( x, y, fout ) # 最小2乗近似


def least_square(x: Dvector, y: Dvector, fout):
    global M, N
    p = Dmatrix(1, N+1, 1, N+1) # p[1...N+1][1...N+1]
    a = Dvector(1, N+1)         # a[1...N-1]

    # 右辺ベクトルの作成
    for i in range(1, N+2):
        a[i] = 0.0
        for j in range(1, M+1):
            a[i] += y[j] * (x[j] ** (i - 1))

    # 係数行列の作成
    for i in range(1, N+2):
        for j in range(1, i+1):
            p[i][j] = 0.0
            for k in range(1, M+1):
                p[i][j] += x[k] ** (i+j-2)
            p[j][i] = p[i][j]

    # 連立一次方程式を解く．結果は a に上書き
    a = gauss2( p, a, N+1 )

    # 結果の出力
    fout.write("最小2乗近似式は y=\n")
    for i in range(N+1, 0, -1):
        fout.write("+ {:5.2f} x^{} ".format(a[i], i-1))
    fout.write("\n")


# 部分ピボット選択付きガウス消去法
def gauss2(a: Dvector, b: Dvector, n: int):
    eps = 2.0 ** -50.0 # eps = 2^{-50} とする

    for k in range(1, n):
        # ピボットの選択
        amax = abs(a[k][k])
        ip = k
        for i in range(k+1, n+1):
            if abs(a[i][k]) > amax:
                amax = abs(a[i][k])
                ip = i

        # 正則性の判定
        if amax < eps:
            print("入力した行列は正則ではない!!!")
        # 行交換
        if ip != k:
            for j in range(k, n+1):
                tmp = a[k][j]
                a[k][j] = a[ip][j]
                a[ip][j] = tmp
            tmp = b[k]
            b[k] = b[ip]
            b[ip] = tmp

        # 前進消去
        for i in range(k+1, n+1):
            alpha = -a[i][k] / a[k][k]
            for j in range(k+1, n+1):
                a[i][j] = a[i][j] + alpha * a[k][j]
            b[i] = b[i] + alpha * b[k]

    # 後退代入
    b[n] = b[n] / a[n][n]
    for k in range(n-1, 0, -1):
        tmp = b[k]
        for j in range(k+1, n+1):
            tmp = tmp - a[k][j] * b[j]
        b[k] = tmp / a[k][k]

    return b


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