Evaluating Different Algorithms for Solving Least Square Problems in Chemical Mass Balance Model

Wei Fang
Aerospace Engineering
MS 2nd year

Yining Chen
Aerospace Engineering
PhD 4+th year

Ziyan Wu
Civil and Environmental Engineering
MS 1st year

Zongrun Li
Civil and Environmental Engineering
MS 1st year

1 ABSTRACT

Chemical mass balance (CMB) receptor model is widely used to quantify different source contributions to the receptor. CMB model is based on solving a series of linear equations for a non-negative result. Our project give a mathematics expression for CMB problem by using matrix. Then we base on active-set method to solve the non-negative least square (NNLS) problem. Additionally, we prove that some steps in the active-set algorithm are equal to solve a full rank least square problem. Based on the proof, we design algorithms by combing householder method, givens method, classical Gram-Schmidt method, modified gram-Schmidt method and normal equation method with active-set method. Furthermore, we discuss these methods' accuracy and efficiency.

2 INTRODUCTION

Particulate matter (PM) causes serious health problems and it also affect visibility. Nowadays, people more and more concern about PM especially in developing countries. Since that, knowing sources and their contributions for PM is important. Chemical mass balance (CMB) receptor model is a common tool for apportioning of ambient levels of pollutants and the core of the model is solving a series of non-negative least square problems. So, this project studies the use of non-negative linear algebra method and combines non-negative least square method with least square method including Householder method, Givens method, the normal equation and the classical and modified Gram-Schmidt method on the application of CMB. After applying each method to an obtained public dataset, both accuracy and efficiency of different methods are compared with each other to illustrate the behavior of non-negative least square problem solving with numerical linear algebra.

3 LITERATURE REVIEW

There are many algorithms based on basis least square algorithms. Meng Dawn Cheng and Philip K Hopke combines Householder and Givens QR method[6]. This combining method is intending to improve the computation speed while remaining an acceptable accuracy. Götze, Jürgen and Schwiegelshohn find a algorithm based on Givens QR to avoid calculating square root which is a time-consuming step[2]. And for non-negative least square problem, the active-set method which is published by Lawson and Hanson is widely used[3]. And Dandong Wang et al. also evaluated different non-negative least square algorithms by solving CMB models[7].

Ordinary weighted least squares (OWLS) method takes the problem of unequal reliability of elements used in the least-squares fitting scheme into consideration, thereby, includes the uncertainties of the measured ambient concentrations, e, and has been used extensively.

However, the uncertainties will affect the solution of the least-squares calculation. So, the OWLS method may underestimate the uncertainties of the whole system of the CMB model by including only the source composition uncertainties. The sufficient variance weighted least squares (EVWLS) have been proposed to solve the problem by measuring uncertainties of both source composition and ambient concentration. Eventually, the uncertainty of the source composition is propagated through the iterative-reweighted procedure to produce the final solution for the CMB receptor model[1].

In general, the householder method is the most robust while other methods such as Givens, Gram-Schmidt is easier to fail. Normal equation is more computationally costly than other methods. So, in this project, we check whether the methods will fail for our input data to show the robustness of each method.

4 PROBLEM STATEMENT

The CMB model is often used to analyze the source apportionment of air pollutants in environmental study. CMB used both chemical and physical characteristics of particles at sources and receptors to find major sources of particle matter[5]. The model is based on following assumptions[4]: (i) Chemical species do not react with each other which means we can add them linearly. (ii) All sources which will significantly contribute to the receptor need be considered in the model. (iii) The source compositions are linearly independent. (iv) The number of sources are less than or equal to the number of species. (v) Measurement uncertainties should be random and have normal distribution. (vi) Compositions of source emissions will not be changed during the measurement.

Using these assumptions, we can use chemical and mass balance to build a series linear equations:

$$C_i = \sum_{j=1}^{n} f_{ij} S_j + e_i$$

Where C_i is the ambient concentration of species i, f_{ij} is the fraction of species i from source j. S_j is the strength of source j.

We can use following equation to describe the series of linear equations if we have m species:

$$C = FS + e$$

Where $C \in \mathbb{R}^m$, $F \in \mathbb{R}^{m \times n}$, $S \in \mathbb{R}^n$ and $e \in \mathbb{R}^m$. As the assumption, we know $m > n$ and our target is to solve this equation for S to minimize $||C - FS||$. If we consider uncertainties in the ambient measurements and ambient profile, the problem will be equivalent to minimize χ^2, where[8]

$$\chi^2 = \sum_{i=1}^{m} \frac{(C_i - \sum_{j=1}^{n} f_{ij} S_j)^2}{\sigma_{C_i}^2 + \sum_{j=1}^{n} \sigma_{f_{ij}}^2 S_j^2}$$

In this project, we will only consider uncertainties in the ambient measurements. Since that, $\sigma_{f_{ij}}$ is 0. The problem can be described as:

$$||\sigma(C - FS)||_{min} = ||(\sigma F)S - (\sigma C)||_{min}$$

Where

$$\sigma = diag(\frac{1}{\sigma_{C_1}}, \frac{1}{\sigma_{C_2}} .... \frac{1}{\sigma_{C_n}}) \in \mathbb{R}^{m \times m}$$

Let $\sigma F = A \in \mathbb{R}^{m \times n}$ and $\sigma C = b \in \mathbb{R}^m$, the problem is equivalent to we need to find a column vector $S \in \mathbb{R}^n$ to minimize χ, where

$$\chi = ||AS - b||$$

For pursuing meaningful results, we also need to make sure all results are non-negative. Obviously, it is a non-negative least square problem. In order to solve a large set of data in CMB, efficiency and accuracy numerical linear algebra methods are required for computation.

5 METHODS

5.1 NNLS algorithm

We use active-set method to solve the non-negative least square problem[3].

Algorithm 1 Non-negative Least Square

```
Input: $E \in \mathbb{R}^{m \times n}$ $f \in \mathbb{R}^m$
Output: $x \in \mathbb{R}^n$
 1: set $\mathcal{P}$=null, $\mathcal{Z} = \{1, 2, 3, 4...n\}$, $x = 0$
 2: Compute the n-vector $w = E^T(x - Ex)$
 3: If set $\mathcal{Z}$ is empty or if $w_j \leq 0$ for all $\mathcal{Z}$, go to Step 12.
 4: Find an index $t \in \mathcal{Z}$ such that $w_t = max\{w_j : j \in \mathcal{Z}\}$.
 5: Move the index $t$ from set $\mathcal{Z}$ to $\mathcal{P}$.
 6: Let $E_{\mathcal{P}}$ denote the $m \times n$ matrix defined by Column of $E_{\mathcal{P}} = \begin{cases} \text{column j of E} & \text{j} \in \mathcal{P} \\ 0 & \text{j} \in \mathcal{Z} \end{cases}$ Compute the n-vector x as a solution of the least square problem $E_{\mathcal{P}} z \approx f$. Only components $z_j, j \in \mathcal{P}$ are determined by this problem. For $z \in \mathcal{Z}$, define $z_j = 0$.
 7: If $z_j > 0$ for all $j \in \mathcal{P}$, set $x = z$ and go to Step 2.
 8: Find an index $q \in \mathcal{P}$, st. $x_q/(x_q - z_q) = min\{x_j/(x_j - z_j)$ for $\text{z}_j \leq 0, j \in \mathcal{P}\}$
 9: Set $\alpha = x_q/(x_q - z_q)$
10: Set $x = x + \alpha(z - x)$
11: Move form set $\mathcal{P}$ to set $\mathcal{Z}$ for $x_j = 0$ where $j \in \mathcal{P}$. Go to Step 6.
12: The computation is completed.
```

5.2 NNLS Combined with Different LS Algoritohms

Algorithm 1 will produce a series of $E_{\mathcal{P}}$. Some of columns in these matrices are zero vectors and the matrices are rank deficient. We will prove that solving least square problem $E_{\mathcal{P}} z \approx f$ and generating a special vector z (step 6) is equivalent to solving a full rank least square problem.

Proof. First, it is obvious that if a matrix has a zero vector in nth column, the nth element is free to choose. Then let $\mathcal{P} = \{p_1, p_2, p_3....p_k\}$ and $\mathcal{Z} = \{z_1, z_2, z_3...z_{n-k}\}$. Without loss of generality , let $p_1 < p_2 < ... < p_k$ and $z_1 < z_2 < ... < z_{n-k}$. We know z_{ith} column is zero vector, for $i \in [1, n-k]$. So, z_{ith} element for z is free to choose and let them as 0 which achieves the Step 6's condition. Then we use permutation matrix to permute $E_{\mathcal{P}}$. We start check the column with the smallest column numbers. When we check the i^th column, if $i \in \mathcal{Z}$, we permute it with the smallest number p_k which $p_k > i$. If $i \in \mathcal{P}$, let i = i+1. After that, $E_{\mathcal{P}}\Pi\Pi^{-1}z = \begin{bmatrix} A & 0 \end{bmatrix} (\Pi^{-1}z)$.

Now, for $A \in \mathbb{R}^{k \times n}$, the column is i^{th} column of $E_{\mathcal{P}}$ where $i \in \mathcal{P}$. For $\Pi^{-1}z$, it does the same transform for the elements which means the element is i^{th} element of z where $i \in \mathcal{P}$. The problem is equivalent to:

$$|| \begin{bmatrix} A & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ 0 \end{bmatrix} - f || = || \begin{bmatrix} A & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ 0 \end{bmatrix} - \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} || = || \begin{bmatrix} Az_1 - f_1 \\ -f_2 \end{bmatrix} ||$$

The least square problem is equivalent to solve $||Az_1 - f_1||$ which is a full rank least square problem. □

5.3 Data Source

The data is the PM2.5 data from North Carolina during Jan 1st to Jan 31st in 2002. Thanks for professor Russell for providing this data for us.

6 THE EXPERIMENT SETTING AND RESULT

The CMB model is used to find out each contribution of sources to the PM2.5. Since each source emits a variety of chemicals and each chemical contribute differently to the PM value, it is a bi-linear model we need to solve. Moreover, since all the assumptions we have in the problem statement section, the solutions of the least square problem we need to solve have a constraint, namely each entry must be positive.

During solving the non-negative least square problem, the Householder QR, Givens QR, normal equations, and the classical and modified Gram-Schmidt reduced QR decomposition algorithms are implemented in MATLAB.

The first method applied in the CMB is householder decomposition with partial pivoting. Compared to standard householder method, it was modified to including row pivoting. This modification is made in order to increase the magnitude of vector "v" in the householder vector, as the equation shown below.

$$P = I_n - \frac{2vv^T}{v^Tv} = I_n - 2\frac{v}{||v||_2} \cdot \frac{v^T}{||v||_2} \qquad where \quad v \neq 0 \in \mathbb{R}^n$$

$$v = x \pm ||x||_2 e_1$$

Because a typical data set of environmental study with CMB includes a lot of zeros and very small values. The magnitude of vector "v" is often very small or close to zero. Hence, When the algorithm goes through the decomposition column by column, if the leading element of column vector x is zero, the algorithm will permute the largest element to the leading position, so the magnitude of v is increased. Then, the householder decomposition is performed.

Since the matrix formed by the measurement data has a lot of zeros, the standard Givens method was modified to avoid dividing by zero issue. The typical Givens matrix is as follows:

$$G(\theta)^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} * \\ -sx_1 + cx_2 \end{bmatrix}$$

$$sx_1 = cx_2 \longrightarrow \frac{s}{c} = \frac{x_2}{x_1} = tan\theta$$

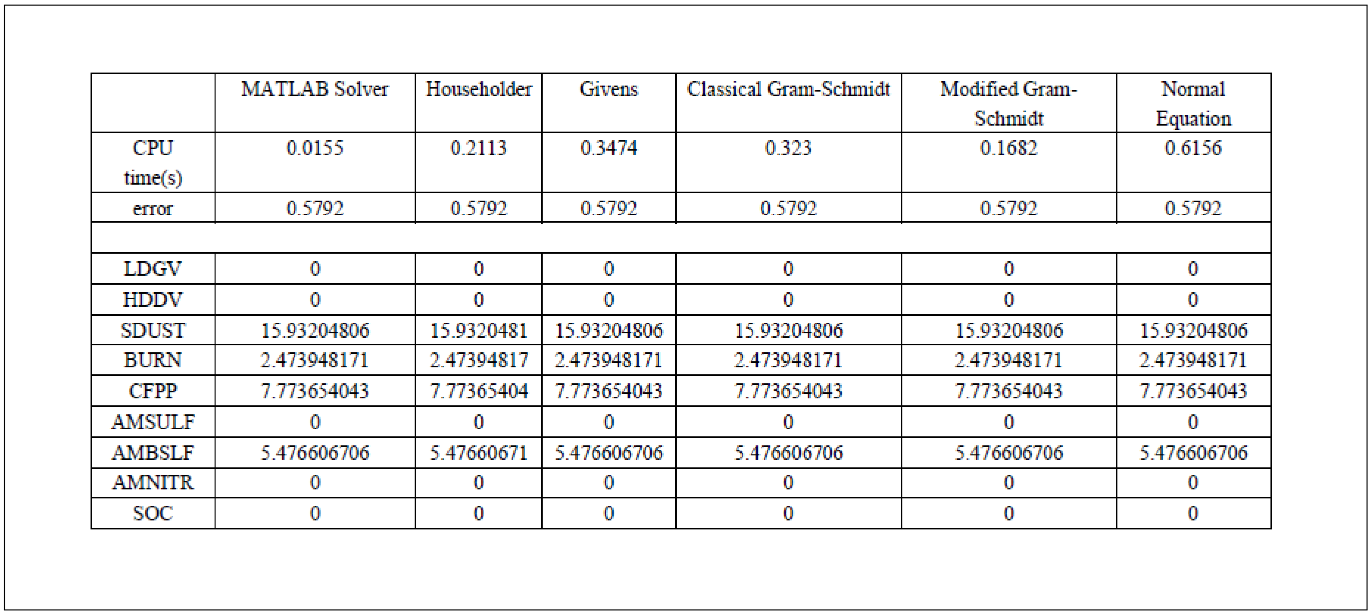
It can not handle zeros. In order to avoid this problem, the algorithm added a checking statement before apply Givens rotation. When there is a dividing zero case occurs, it will simply exchange the rows.The Givens matrix handle the zeros is as follows:

$$G = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The third and fourth methods applied are Classical and Modified Gram-Schmidt. Both methods have accurate results compare to MATLAB solver. However, the Modified Gram-Schmidt took less computational time than Classical Gram-Schmidt method.

Normal equation method is also applied in the CMB model, the results shown that the normal equation method took the longest computational time compare to other methods. The obtained input data has a condition number in $O(10^4)$, which is an acceptable value comparing to the machine precision in $O(10^{-16})$, but it is also not a good condition number.

In the table 1 are the results got by Householder, Givens, Classical Gram-Schmidt,Modified Gram-Schmidt and normal equation method. The results match well with the built-in MATLAB solver for non-negative least square problem. As expected, Givens and normal equations use the longest time as in the QR factorization process their computational cost is higher.



	MATLAB Solver	Householder	Givens	Classical Gram-Schmidt	Modified Gram-Schmidt	Normal Equation
CPU time(s)	0.0155	0.2113	0.3474	0.323	0.1682	0.6156
error	0.5792	0.5792	0.5792	0.5792	0.5792	0.5792
LDGV	0	0	0	0	0	0
HDDV	0	0	0	0	0	0
SDUST	15.93204806	15.9320481	15.93204806	15.93204806	15.93204806	15.93204806
BURN	2.473948171	2.47394817	2.473948171	2.473948171	2.473948171	2.473948171
CFPP	7.773654043	7.77365404	7.773654043	7.773654043	7.773654043	7.773654043
AMSULF	0	0	0	0	0	0
AMBSLF	5.476606706	5.47660671	5.476606706	5.476606706	5.476606706	5.476606706
AMNITR	0	0	0	0	0	0
SOC	0	0	0	0	0	0

Table 1: Result

We also tried some other data to our code. Due to specific property of the specific data set, some algorithm can't solve the problem. For example, when we combine nnls with householder method, we need to set a tolerance at step 3 ($w_j \leq \epsilon$, where $\epsilon > 0$) to find a numerical result. Thus, the robustness of each method depends on the problem properties. So choose proper method is important in real applications.

7 CONCLUSION

7.1 Algorithms

By modifying non-negative least square algorithm, we combine different least square algorithms with non-negative least square algorithm. We use Householder, Givens, normal equations, and the classical and modified Gram-Schmidt method and compare their performance. For the householder methods, We added row pivoting before each householder vector calculation to increase the magnitude of "v". In the Givens implementation, we add a check statement to see whether we will encounter dividing zero situations. If that happens, we make the Givens matrix to be a permutation matrix to switch corresponding rows to avoid this situation. This guarantees the code will work for our input.

7.2 Source Contributions

According to the Figure 1 Different source of pollution changed by date, we can see that in the dates in January 2002, the LGDV (light-duty diesel vehicles) has several peaks in the first half of January 2002, the HDDV (heavy-duty diesel vehicle) has several almost same peaks in the first half of January 2002 with one-tenth the value of LGDV. The SDUST (soil dust) has several peaks in the second half of January 2002. BURN (biomass burning) was the most common pollution source in January 2002. AMSULF (ammonium sulfate) is one of the crucial pollution sources and has several peaks thorough the whole month. AMNTIR (ammonium bisulfate) was decreasing during the whole week, but it was still one of the most common pollution sources in January 2002. SOC had three waves almost evenly distributed in mid-January 2002.

Overall, there were three concentration peaks at the beginning, the middle and end of January 2002.

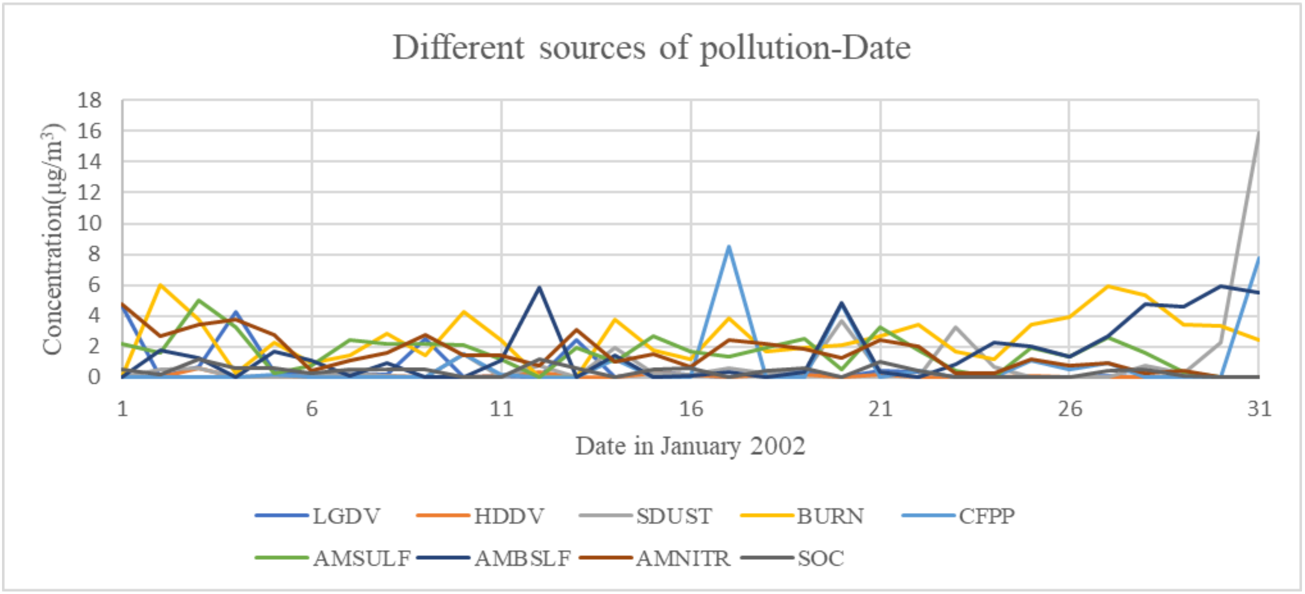


Figure 1: Different Sources VS Time

From figure 2, we find that among all nine individual sources of pollution, BURN is the most prevalent one, which occupies 24% of the total pollution in January 2002. Also, LGDV, AMNITR, and AMSULF are the common-seen pollution source.

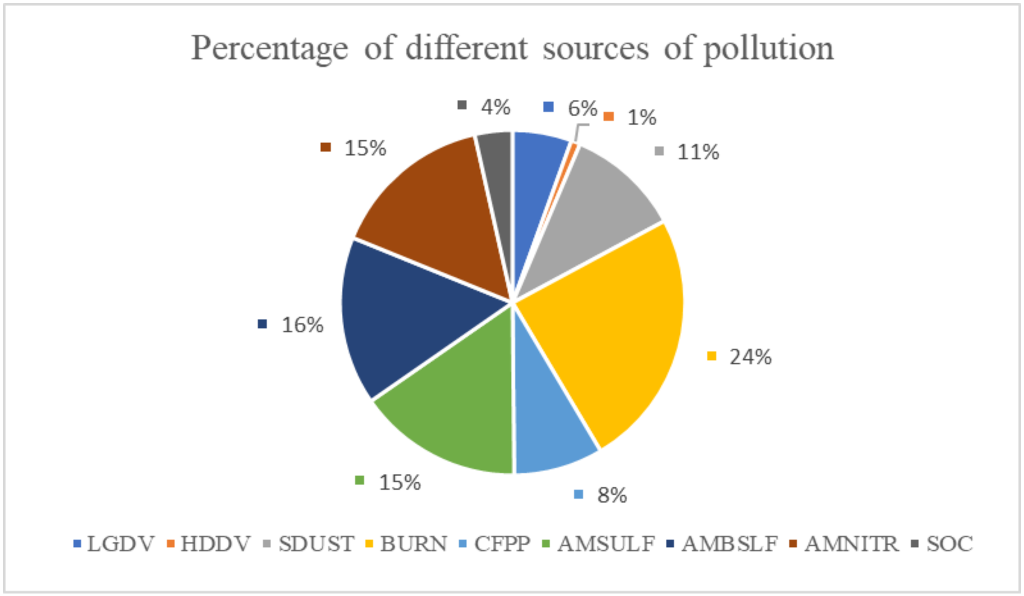


Figure 2: Contributions of Sources

In summary, by comparison of different methods to solve the non-negative least square problem in CMB model, we find the percentage of different sources of pollution and its variation by date. Among the five methods we use, the Householder and Modified Gram-Schmidt are the fastest for the specific data we use. Pivoting methods are used in the Householder and Givens to deal with the zeros in the solving of the non-negative least square algorithm.

REFERENCES

[1] Meng Dawn Cheng and Philip K Hopke. 1986. Investigation on the use of chemical mass balance receptor model: numerical computations. *Chemometrics and intelligent laboratory systems* 1, 1 (1986), 33–44.
[2] Jürgen Götze and Uwe Schwiegelshohn. 1991. A square root and division free Givens rotation for solving least squares problems on systolic arrays. *SIAM J. Sci. Statist. Comput.* 12, 4 (1991), 800–807.
[3] Charles L Lawson and Richard J Hanson. 1995. *Solving least squares problems*. Vol. 15. Siam.
[4] Amit Marmur, Alper Unal, James A Mulholland, and Armistead G Russell. 2005. Optimization-based source apportionment of PM2. 5 incorporating gas-to-particle ratios. *Environmental science & technology* 39, 9 (2005), 3245–3254.
[5] Marco Martínez-Cinco, Jesús Santos-Guzmán, and Gerardo Mejía-Velázquez. 2016. Source apportionment of PM2. 5 for supporting control strategies in the Monterrey Metropolitan Area, Mexico. *Journal of the Air & Waste Management Association* 66, 6 (2016), 631–642.
[6] Alex Pothen and Padma Raghavan. 1989. Distributed orthogonal factorization: Givens and Householder algorithms. *SIAM J. Sci. Statist. Comput.* 10, 6 (1989), 1113–1134.
[7] Dandong Wang and Philip K Hopke. 1989. The use of constrained least-squares to solve the chemical mass balance problem. *Atmospheric Environment (1967)* 23, 10 (1989), 2143–2150.
[8] John G Watson, John A Cooper, and James J Huntzicker. 1984. The effective variance weighting for least squares calculations applied to the mass balance receptor model. *Atmospheric Environment (1967)* 18, 7 (1984), 1347–1355.

A CODE

A.1 main

```
clear;
clc;
% C:\Users\dell\Desktop\Numerical\project\input.xls
path = input('Input path of CMB data: ','s');
concentrationData = readtable(path,'sheet',1);
[row1,column1] = size(concentrationData);
concentration = table2array(concentrationData(1:row1,3:column1));

uncertaintyData = readtable(path,'sheet',2);
[row2,column2] = size(uncertaintyData);
sigmaConcentration = table2array(uncertaintyData(1:row2,3:column2));

profileData = readtable(path,'sheet',3);
[row3,column3] = size(profileData);
profile = table2array(profileData(1:row3,2:column3));

[row4, column4] = size(concentration);
% calculate sigma sigma*F sigma*C
% row4 = 3; % CHANGE THERE NOT USE ALL DATA TO TEST!!!
T = zeros(1, row4);
MyResult = [];
error = [];
for i = 1:row4
    % sigmaConcentration(i,:) C = concentration(i,:)
    sigma = zeros(column4);
    for j = 1:column4
        sigma(j,j) = sigmaConcentration(i,j);
    end
    F = profile';
    C = concentration (i,:)';
    A = sigma*F;
    b = sigma*C;
    % calculate ||Ax - b||min
    tic
    % A
    % b
    %[x] = lsqnonneg(A,b);
    [x]=nnls2(A,b);
    MyResult = [MyResult,x];
    e = norm(A*x - b,2);
    error = [error,e];
    T(i) = toc;
end
tMul = sum(T)
norm(error)
```

A.2 None Negative Least Square Solver

```
function [x] = nnls2(E,f)
% step 1:
[m,n]=size(E);
Z = 1:1:n;
P = [];
Universe = 1:1:n;
x = zeros(n,1);
% step 2
w = E'*(f-E*x);
while (true)
    % j in Z, w_Z
    w_Z = w(Z);
    % step 3
    if(isempty(Z) || max(w_Z) ≤0)
        break;
    else
        % find t step 4
        t = Z(w_Z==max(w_Z));
        % step 5
        Z(Z==t) = [];
        P = setdiff(Universe, Z);
        % step 6
        while (true)
            E_p = zeros(m,length(P));
            for column= 1:length(P)
                E_p(:,column)= E(:,P(column));
            end
            % compute least square step 6
            % Solve full rank least square problem here!!!
            % z_test = E_p\f;
            % z_test=givens(E_p,f);
            % z_test=classicalgs(E_p,f);
            % z_test=householder(E_p,f);
            z_test=modifiedgs(E_p,f);
            % z_test=normaleqn(E_p,f);
            z = zeros(n,1);
            for i = 1:length(P)
                z(P(i)) = z_test(i);
            end
            % step 7
            z_P = z(P);
            if min(z_P) > 0
                break;
            else
                % step 8
                alphaArray = [];
                for element = P
                    if z(element)≤0
                        alphaArray = [alphaArray,x (element)/(x(element)-z(element))];
                    end
                end
                % step 9
                alpha = min(alphaArray);
                % step 10
                x = x + alpha.*(z-x);
                for j = P
                    if x(j) == 0
                        Z = [Z,j];
                    end
                end
                P = setdiff(Universe, Z);
            end
        end
        x = z;
        w = E'*(f-E*x);
    end
end
end
```

A.3 Householder Method

```
function [x_house] = householder(A,b)
[m,n] = size(A);

% generated a [0;L] matrix with householder
for i = 0:n-1
    x = A(1:m-i,n-i);
    v = x;
    ind = eye(length(v));

    % pivoting
    if v(end) == 0
        [x_max,x_ind] = max(v);
        inx = 1:length(v);
        inx(end) = x_ind;
        inx(x_ind) = length(v);
        ind = ind(inx,:);
    end
    ind_new = blkdiag(ind,eye(i));
    A = ind_new*A;
    b = ind_new*b;
    x = A(1:m-i,n-i);
    v = x;

    % household vector
    v(end) = v(end) + sign(v(end))*norm(x,2);
    p = eye(m);
    p(1:m-i,1:m-i) = eye(length(v)) - 2.*v*v'./(v'*v);
    A = p*A;
    b = p*b;
end

L = A(m-n+1:m,:);
x_house = L\b(m-n+1:end);
end
```

A.4 Givens Method

```
function [x_givens] = givens(A,b)

[m,n] = size(A);

% construct givens method
for q = 2:m
    for p = 1:min(q-1,n)
        if A (p,p)≠0
        t = A (q,p)/A(p,p);
        c = 1/sqrt(1+t^2);
        s = c*t;
        J = eye(m);
        J(p,p) = c;
        J(q,q) = c;
        J(p,q) = s;
        J(q,p) = -1*s;
        else
        J = eye(m);
        J(p,p) = 0;
        J(q,q) = 0;
        J(p,q) = 1;
        J(q,p) = 1;
        end
        A = J*A;
        b = J*b;
    end
end

R = A(1:n,:);
% output
uptriang = dsp.UpperTriangularSolver;
% x_givens = R\b(1:size(A,2));
x_givens = uptriang(R,b(1:size(A,2)));
end
```

A.5 the classical Gram-Schmidt method

```
function [x_cgs] = classicalgs(A,b)

[m,n] = size(A);
L(n,n) = norm(A(:,n));
Q1(:,n) = A (:,n)/L(n,n);

% generated Q1 and L with classical gram-schmidt
for k = n-1:-1:1
    sum_rq = 0;
    for i = n:-1:k+1
        L(i,k) = Q1 (:,i)'*A(:,k);
        sum_rq = sum_rq + L(i,k)*Q1(:,i);
    end
    z(:,k) = A(:,k) - sum_rq;
    L(k,k) = norm(z(:,k));
    Q1(:,k) = z (:,k)/L(k,k);
end

% output
lowtriang = dsp.LowerTriangularSolver;
x_cgs = lowtriang(L,Q1'*b);

end
```

A.6 modified Gram-Schmidt method

```
function [x_mgs] = modifiedgs(A,b)

[m,n] = size(A);
L = eye(n);
Q1 = eye(n);
% generated Q1 and L with modified gram-schmidt
for k = n:-1:1
    L(k,k) = sqrt(sum(A (:,k).^2));
    for i = 1:m
        A(i,k) = A (i,k)./L(k,k);
    end
    for j = 1 :k-1
        L(k,j) = sum(A (:,k).*A(:,j));
        for i = 1:m
            A(i,j) = A(i,j) - A(i,k)*L(k,j);
        end
    end
end
Q1 = A;
d = Q1'*b;
lowtriang = dsp.LowerTriangularSolver;
x_mgs = lowtriang(L,d);

end
```

A.7 normal equation Method

```
function [x_ne] = normaleqn(A,b)

C = A'*A;
d = A'*b;
R = chol(C);
lowtriang = dsp.LowerTriangularSolver;
uptriang = dsp.UpperTriangularSolver;
y = lowtriang(R',d);
x_ne = uptriang(R,y);

end
```