Incomplete Logarithmic Least Squares Method using **fmincon**

Hailemariam A. Tekile

06 January 2023

Incomplete LLSM using **fmincon** on the interval [1/9,9]. i.e, 1/9<=w(i)/w(j)<=9 , to put inside fmincon

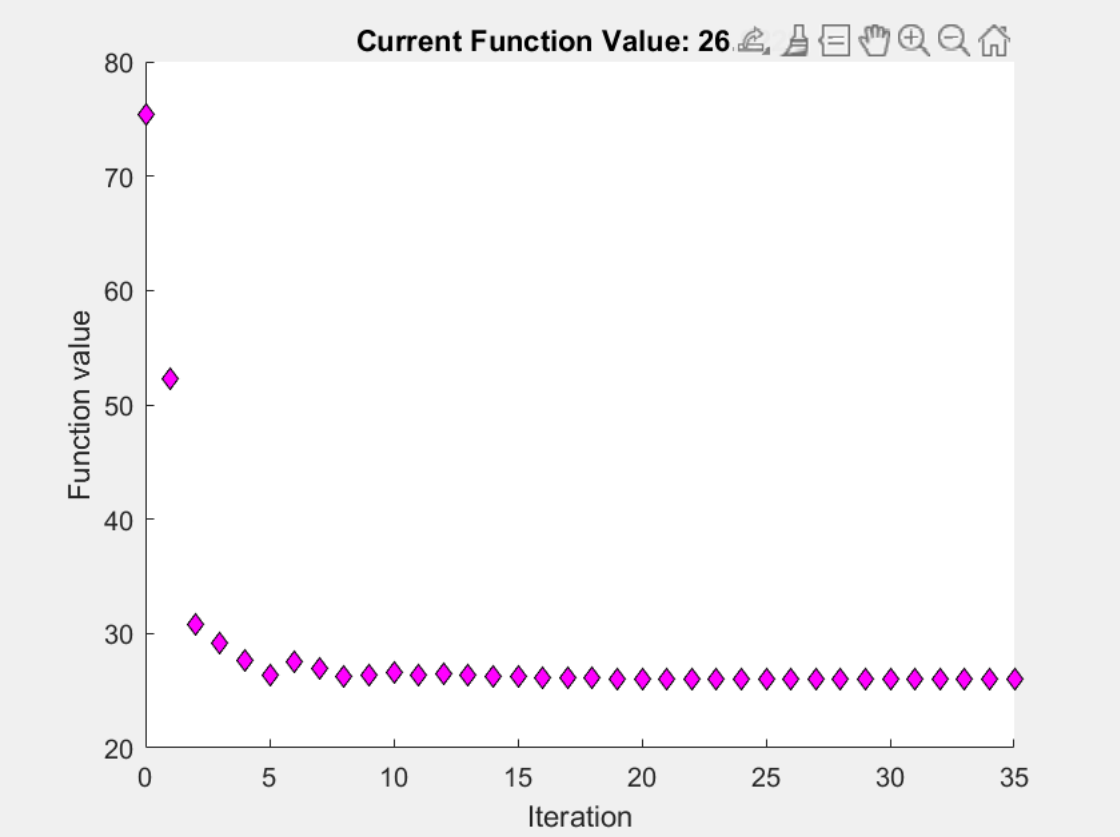
The constraints are:

1. 1/9 <=x(i)/x(j)<=9, for i,j=1,2,3,4,...,n by considering i<j.
2. x(1) + x(2) + x(3) + x(4) +...+x(n)= 1;
3. x(i) > 0 for i=1,2,3,4,...,n

Reference: Tekile, H. A., Brunelli, M., & Fedrizzi, M. (2023). A numerical comparative study of completion methods for pairwise comparison matrices. *Operations Research Perspectives*, 100272.

```
clear; close; clc

% global variables
global A;
global numberOfMissingEntries;
global wPosition;
global n;

% Incomplete matrix, 0 representes the missing entries
A =[...
    1   3   1/2 2 1/9   2 1/2 2;...
    1/3 1   4   1 1/4 1 1/4 1;...
    2   1/4 1   4 1   4 1   4;...
    1/2 1   1/4 1 1/4 8 1/4 1;...
    9   4   1   4 1   4 1   4;...
    1/2 1   1/4 1/8 1/4 1 1/4 1/7;...
    2   4   1   4 1   4 1   0;...
    1/2 1   1/4 1 1/4 7 0   1];

%number of missing entries in the upper triangular matrix
numberOfMissingEntries = 1;
n = size(A,1);

w0=1/9*ones(1,n); %initial value


% w_i>0, but not on [1/9,9]
 lb=0.0001*ones(1,n); % lower bound
 ub=[];   % upper bound

 % % normalize w inside fmincon
 Aeq = ones(1,n); % as w(1)+...+w(n) = 1
 beq = 1;
```

```
% using Matlab's fmincon
% w= fmincon(@objFunction,w0,[],[],Aeq,beq,lb,ub,@mycon)
options = optimoptions('fmincon','Algorithm','interior-point','ConstraintTolerance',1e-15,'Plot
w= fmincon(@objFunction,w0,[],[],Aeq,beq,lb,ub,@mycon,options)
```




```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

<stopping criteria details>
w = 1×8
    0.0979    0.0755    0.1588    0.0728    0.2710    0.0339    0.2156    0.0746
```

```
%
% Alternative normalization of w, out of fmincon
% w_sum = sum(w);
% w_vector = w/w_sum

% to write the  completed PCM
for r4 = 1:numberOfMissingEntries % r4 is another index
    A(wPosition(r4,1),wPosition(r4,2)) = w(wPosition(r4,1))/w(wPosition(r4,2)); % equivalent to
    A(wPosition(r4,2),wPosition(r4,1)) = w(wPosition(r4,2))/w(wPosition(r4,1)); % equivalent to
end
 disp('completed matrix with in the interval [1/9,9]')
```

```
completed matrix with in the interval [1/9,9]
```

```
A % the completed PCM
```

```
A = 8×8
    1.0000    3.0000    0.5000    2.0000    0.1111    2.0000    0.5000    2.0000
    0.3333    1.0000    4.0000    1.0000    0.2500    1.0000    0.2500    1.0000
    2.0000    0.2500    1.0000    4.0000    1.0000    4.0000    1.0000    4.0000
    0.5000    1.0000    0.2500    1.0000    0.2500    8.0000    0.2500    1.0000
    9.0000    4.0000    1.0000    4.0000    1.0000    4.0000    1.0000    4.0000
    0.5000    1.0000    0.2500    0.1250    0.2500    1.0000    0.2500    0.1429
    2.0000    4.0000    1.0000    4.0000    1.0000    4.0000    1.0000    2.8921
    0.5000    1.0000    0.2500    1.0000    0.2500    7.0000    0.3458    1.0000
```

```
% matrix DD based on the obtained w(i)/w(j)
DD = zeros(n);
for i=1:n-1
    for j=i+1:n
        DD(i,j) = w(i)/w(j);
        DD(j,i) = w(j)/w(i);
    end
end
DD;
%diagonals are all 1s
for i=1:n
    DD(i,i)=1;
end
disp('completed matrix from w(i)/w(j)')
```

```
completed matrix from w(i)/w(j)
```

```
DD % completed matrix from w(i)/w(j)
```

```
DD = 8×8
    1.0000    1.2968    0.6164    1.3443    0.3611    2.8835    0.4539    1.3126
    0.7711    1.0000    0.4753    1.0366    0.2785    2.2234    0.3500    1.0122
    1.6224    2.1040    1.0000    2.1810    0.5859    4.6781    0.7364    2.1296
    0.7439    0.9647    0.4585    1.0000    0.2686    2.1449    0.3376    0.9764
    2.7690    3.5909    1.7067    3.7224    1.0000    7.9843    1.2568    3.6347
    0.3468    0.4498    0.2138    0.4662    0.1252    1.0000    0.1574    0.4552
    2.2032    2.8573    1.3580    2.9619    0.7957    6.3530    1.0000    2.8921
    0.7618    0.9880    0.4696    1.0241    0.2751    2.1967    0.3458    1.0000
```

```
% normalized right eigenvector by Eigenvector method
[V,D]=eig(A);
eigD=diag(D);
[~,maxindex]=max(eigD);
weight=zeros(n,1);
for i=1:n
    weight(i)=V(i,maxindex);
end
% normalize weights
normalize = sum(weight);
disp('Weight vectors from RIGHT eigenvector method:')
```

```
Weight vectors from RIGHT eigenvector method:
```

```
right_eigenvector = weight*(1/normalize)
```

```
right_eigenvector = 8×1
    0.0986
    0.1063
    0.1569
    0.0746
    0.2657
    0.0364
    0.1886
    0.0728
```

```
%Geometric Mean Method using rows
disp('Weight vectors from geometric mean method:')
```

```
Weight vectors from geometric mean method:
```

```
geometric_mean = (geomean(A')./sum(geomean(A')))'
```

```
geometric_mean = 8×1
    0.0979
    0.0755
    0.1588
    0.0728
    0.2710
    0.0339
    0.2156
    0.0746
```

```
% Calculates the minimal lambdaMax
eigenValue=eig(A); % calculates the eigenvalue of A
lambdaMaxMinimal = max(eigenValue); % Perron-Frobenius Theorem
lambdaMaxMinimal;
% % RI values, where n is from 3 to 11
RI = [0  0  0.5247 0.8816 1.1086 1.2479 1.3417 1.4057 1.4499 1.4854 1.5156];
disp('Inconsistency ratio is:')
```

```
Inconsistency ratio is:
```

```
CR =(((lambdaMaxMinimal)-n)/(n-1))/RI(n); % Saaty's inconsistency ratio
CR
```

```
CR = 0.2044
```

```

```

```
function IncompleteLLSM = objFunction(w) % objective function for FedrizziGiove
global A;
global wPosition;
global numberOfMissingEntries;
global n;
B = A; %Incomplete PCM
%to write the position of w
```

```
index = 1; % an index that verifies the position of i and j
for j = 2:n % for i < j
    for i = 1:j-1
        if B(i,j) == 0
            wPosition(index,1) = i; % position of w in the ith row
            wPosition(index,2) = j; % position of w in the jth column
            index = index+1; % update index until (A(i,j) == 0) ends
        end
    end
end

%
% % conctruct boolean matrix (logical matrix)
Bool = zeros(n,n);
for i=1:n
    for j=1:n
        if B(i,j)==0
            Bool(i,j)=0;
        else
            Bool(i,j)=1;
        end
    end
end
Bool; % Boolean/logical matrix

% Writes w in the missing position
for ss = 1:numberOfMissingEntries % s is another index
    B(wPosition(ss,1),wPosition(ss,2)) = w(wPosition(ss,1))/wPosition(ss,2); % w(i)/w(j)
    B(wPosition(ss,2),wPosition(ss,1)) = w(wPosition(ss,2))/wPosition(ss,1); % w(j)/w(i)

end
B;

% %Objective function
IncompleteLLSM=0;
for i=1:n
    for j=1:n
IncompleteLLSM = IncompleteLLSM + Bool(i,j)*(log(B(i,j))-log(w(i)/w(j))).^2;
    end
end
IncompleteLLSM;

% %Alternative objective function
% %for i<j
% IncompleteLLSM=0;
% for j=2:n
%     for i=1:j-1
% IncompleteLLSM = IncompleteLLSM + Bool(i,j)*((log(B(i,j))-log(w(i)/w(j))).^2 + (log(B(j,i))-l
%     end
% end
```

```
% IncompleteLLSM;

end

%For 1/9<=w(i)/w(j)<=9
function [c,ceq] = mycon(w) % for non linear constraint
global n;
% compute nonlinear inequality constraints
c = zeros(n*(n-1),1); % preallocate, 12 means 2*n*(n-1)/2
m = 0;
for i = 1:n-1 % i=1:n-1 as i<j
    for j = i+1:n % j=i+1:n
        m = m+1;
        c(m) = 1/9 - w(i)/w(j); % the first n(n-1)/2 inequalites
        c(m+n*(n-1)/2) = w(i)/w(j) - 9; % the second n*(n-1)/2 inequalities
    end
end
% compute nonlinear equality constraints at x.
ceq = []; % no nonlinear equality constraints

end
```