Ergu et al.'s 2016 Least Square Method (LSM) with MATLAB'S fmincon

Hailemariam A. Tekile

17 March 2023

The LSM could be used to find the optimal values of missing comparisons by minimizing the sum of errors (geometric mean induced bias matrix (GMIBEM) ε) squares formula:

$$\text{Minimize } f(a_{\text{ij}}, x) = \sum_{j=1}^{n} \sum_{i=1}^{n} \left(\varepsilon_{\text{ij}}(x, a_{\text{ij}})\right)^2,$$

subject to the interval constraint $\left[\frac{1}{9}, 9\right]$.

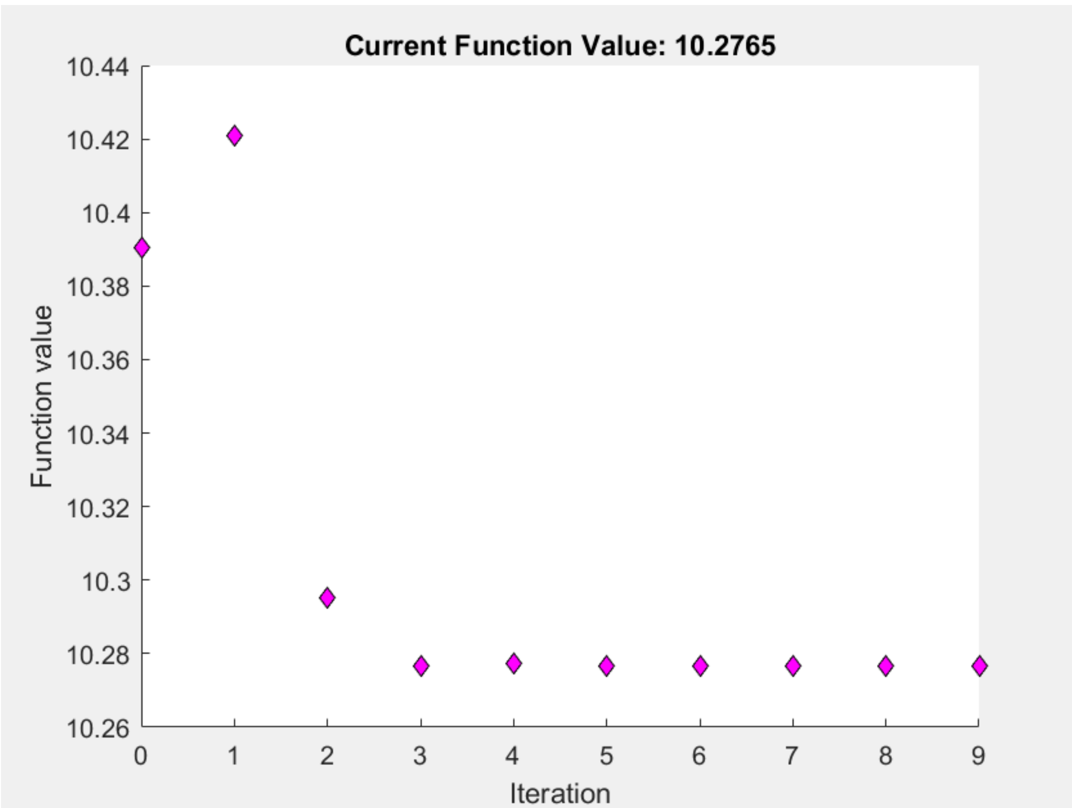
References:

1. Tekile, H. A., Brunelli, M., & Fedrizzi, M. (2023). A numerical comparative study of completion methods for pairwise comparison matrices. *Operations Research Perspectives*, 100272.
2. Ergu, D.; Kou, G.; Peng, Y.; Zhang, M. Estimating the missing values for the incomplete decision matrix and consistency optimization in emergency management. Appl. Math. Model. 2016, 40, 254–267


```
clear; close; clc

% global variables
global A;
global NumberOfMissingEntries;
global n;
global xPosition;


% Incomplete PCM
A=[...
 1   0 3 1;...
 0 1  1/2 4;...
 1/3 2  1 5;...
 1 1/4 1/5 1];

%size of A
n = size(A,2);

% number of missing entries in  the upper triangular  matrix A
NumberOfMissingEntries = 1;

x0 = ones(1,NumberOfMissingEntries); % initial value
lb = 1/9*ones(1,NumberOfMissingEntries); % lower bound
```

```
ub = 9*ones(1,NumberOfMissingEntries); % upper bound

%% MATLAB fmincon - applying the algorithm 'interior-point'
options = optimoptions('fmincon','Display','iter','Algorithm',...
    'interior-point','PlotFcns',@optimplotfval);
x= fmincon(@objectiveFunction,x0, [],[],[],[],lb,ub,[],options)
```

```
                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       2    1.039049e+01    0.000e+00    1.119e+00
    1       5    1.042097e+01    0.000e+00    8.330e-01    5.386e-01
    2       7    1.029509e+01    0.000e+00    2.749e-01    2.092e-01
    3       9    1.027665e+01    0.000e+00    1.047e-01    1.148e-01
    4      11    1.027751e+01    0.000e+00    7.834e-02    3.409e-02
    5      13    1.027651e+01    0.000e+00    1.906e-02    1.914e-02
    6      15    1.027646e+01    0.000e+00    5.193e-04    4.920e-03
    7      17    1.027646e+01    0.000e+00    1.796e-04    1.184e-04
    8      19    1.027646e+01    0.000e+00    1.803e-06    4.198e-05
    9      21    1.027646e+01    0.000e+00    1.796e-08    3.643e-07
```




```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
x = 1.2247
```

```
% to write the reconstructed complete PCM
for r = 1:NumberOfMissingEntries % r is another index
    A(xPosition(r,1),xPosition(r,2)) = x(r); % put x(r) in the position (i,j);
    A(xPosition(r,2),xPosition(r,1)) = 1/x(r); % put 1/x(r) in the position (j,i)
```

```
end

% The Completed matrix
B = A;
disp(B)
```

```
    1.0000    1.2247    3.0000    1.0000
    0.8165    1.0000    0.5000    4.0000
    0.3333    2.0000    1.0000    5.0000
    1.0000    0.2500    0.2000    1.0000
```

```
 function LSM =objectiveFunction(x) % objective function
%
global A;
global xPosition;
global NumberOfMissingEntries;
global n;
%xPosition=ones(NumberMissingEntries,2); % preallocate memory for a variable
                                  % to speed up the algorithm
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 A(i,j) == 0
            xPosition(index,1) = i; % position of t in the ith row
            xPosition(index,2) = j; % position of t in the jth column
            index = index+1; % update index until (A(i,j) == 0) ends
        end
    end
end

%t0 = log(ones(1,numberOfMissing)); % initial value: t0 = log([1,1]) or x0 = (1,1).
% using exponential function x = exp(t)
for s = 1:NumberOfMissingEntries % s is another index
    A(xPosition(s,1),xPosition(s,2)) = x(s); % put x in the position (i,j);
    A(xPosition(s,2),xPosition(s,1)) = 1/x(s); % put 1/x in the position (j,i)
end

% Least square Method - Ergu et al. 2016
AT = A'; % transpose of A
L = ones(n,1);
R = ones(1,n);
for i=1:n
    for j=1:n
        L(i,:)=nthroot(prod(A(i,:)),n); % row geometric mean
        R(:,j)=nthroot(prod(A(:,j)),n); % column geometric mean
    end
end

L; %row geometric mean
R;  % column geometric mean
```

```
P = L*R; % matrix product
ErrorMatrix = P.*AT - 1; % GMIBEM error matrix

% Error square
LSM = 0;
for i = 1:n
    for j = 1:n
  LSM = LSM + ErrorMatrix(i,j).^2; % LSM
 %ErrorSquare = ErrorSquare + abs(ErrorMatrix(i,j)); % LAE

    end
end
LSM;

end
```