Hand Measurements:

```
E1 = [85 65 75 75 60 15 17 16 15];
E2 = [84 65 74 73 60 15 16 15 14];
E3 = [82 63 73 70 52 15 16 16 15];
E4 = [84 64 73 72 52 15 17 16 15];
E5 = [84 65 75 72 60 16 17 16 15];

E = [E1; E2; E3; E4; E5];

chosen_indices = [1, 2, 3, 4];

R_raw = mean(E(chosen_indices, :), 1);
R = round(R_raw, 1);
T = E5;
```

Create database matrix and the test matrix.

Create a database matrix “DB” of reference vectors, and a matrix “TEST” of test vectors. For this purpose, we will use your vectors, and the vectors attached as .mat files with this exercise.

Each column in the matrix DB consists of a reference vector of a different person, and the column index is the ID number of the person to whom the reference vector belongs.

Each column in the matrix TEST is a test vector. The column index is also the ID number of the person to whom the test vector belongs.

```
load DB11a.mat
load T11a.mat

DB11a;
T11a;

R = R(:);
T = T(:);

DB = [R, DB11a];
TEST = [T, T11a];
```

Identification Exercise

Carry out identification for the whole group, i.e. for all people in the test data matrix TEST. Then use the results to:

1) Calculate by hand the function P (k) for k = 1, 2, ...

P (k) indicates the chance (probability) that the correct identity is output in position k of IdLista. Draw the function P(k) that you calculated in a graph.

2) Calculate by hand the function TPIR (M) and draw it in a graph (the CMC curve).

TPIR (M) indicates the chance (probability) that the correct identity is output in the first M positions of the output list. TPIR (M) is obtained by summing up P(k) up to k = M.

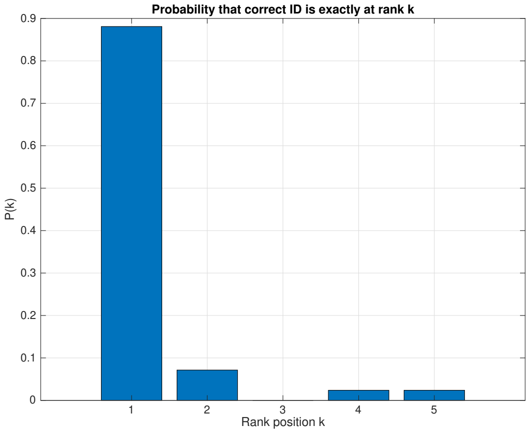
Put also the values of P (k) and TPIR (M) in a table.

3) From TPIR (M), calculate the length of the list M to have a 90% chance that the correct identity is obtained in IdLista among the first M identities.

4) Find the two ID numbers that are the most and the least similar to your own ID. Also write down the distances of the two cases (how similar the hands are).

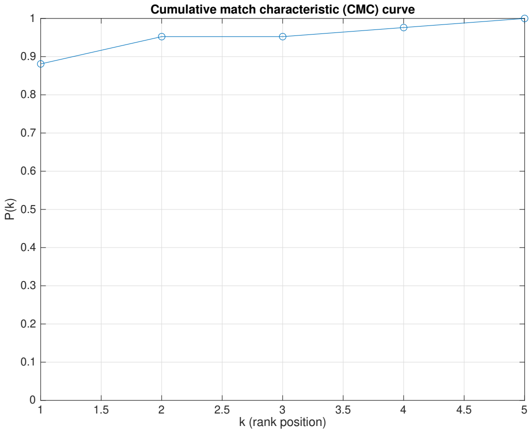
1)

```
Ntests = size(TEST,2);
trueID = 1:Ntests;

rankPos = zeros(1,Ntests);
for j = 1:Ntests
    [IdLista, IdAvst] = MinDistClassID('eucl',DB,TEST,j);
    rankPos(j) = find(IdLista(:,1) == trueID(j), 1, 'first');
end

Kmax = max(rankPos);      % highest rank observed
P = zeros(1,Kmax);

for k = 1:Kmax
    P(k) = sum(rankPos == k) / Ntests;   % fraction of tests at rank k
end

% 3. Plot P(k) as a bar chart
figure;
bar(1:Kmax, P);
xlabel('Rank position k');
ylabel('P(k)');
title('Probability that correct ID is exactly at rank k');
grid on;
print('pk_bar','-depsc');   % Farbiges EPS
```



2)

```
Kmax = max(rankPos);
P = zeros(1,Kmax);
for k = 1:Kmax
    P(k) = sum(rankPos <= k) / Ntests;
end

plot(1:Kmax, P, '-o');
xlabel('k (rank position)');
ylabel('P(k)');
title('Cumulative match characteristic (CMC) curve');
ylim([0,1])
grid on;
print('cmc_curve','-depsc');   % Farbiges EPS
```



3) From TPIR (M), calculate the length of the list M to have a 90% chance that the correct identity is obtained in IdLista among the first M identities.

```
len_M = find(P >= 0.9, 1, 'first')
```

```
len_M = 2
```

4) Find the two ID numbers that are the most and the least similar to your own ID. Also write down the distances of the two cases (how similar the hands are).

```
[IdLista, IdAvst] = MinDistClassID('eucl',DB,TEST,1);
most_sim = IdLista(1, :)
```

```
most_sim = 1
```

```
most_sim_dist = IdAvst(1, :)
```

```
most_sim_dist = 4.4215
```

```
least_sim = IdLista(end, :)
```

```
least_sim = 4
```

```
least_sim_dist = IdAvst(end, :)
```

```
least_sim_dist = 33.1310
```

Verification Mode:

To perform an authentication (one-to-one comparison) it is necessary to determine a value of the distance between the test vector and the reference vector which can be accepted as "sufficiently similar". This value is often called *threshold*.

The attached Matlab function **MinDistClassVER('eucl',DB,TEST,id,Th)** carries out the verification of the persons specified in the input parameter "id" using the threshold "Th".

```
[VerId, VerAvst] = MinDistClassVER('eucl',DB,TEST,1,5) % test
```

```
VerId = 0
VerAvst = 4.4215
```

In order to estimate the error of the system, we should compute statistics of the output distances when hands from the same person and from different persons are compared. This can be done with the provided function: **[ShDist,OhDist]=DistNew('eucl',DB,TEST).**

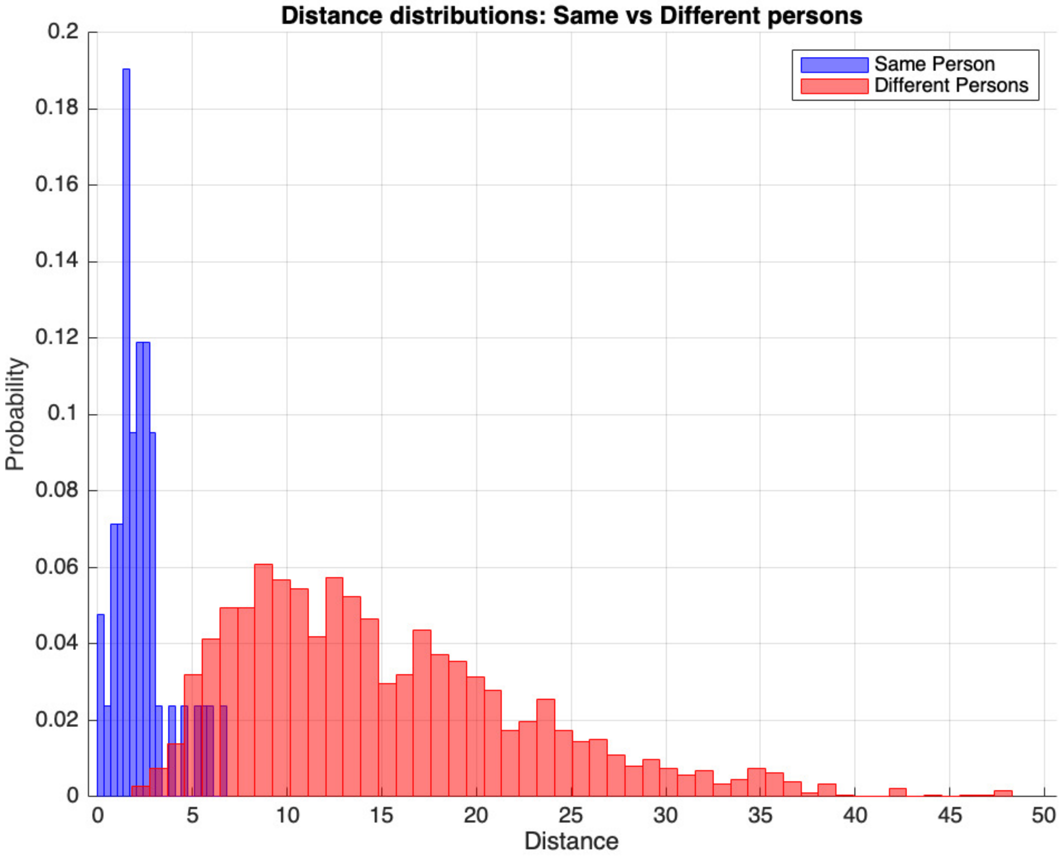
Input parameters DB and TEST are the database data and test data.

Output parameter ShDist contains the distances when hands from the same person are compared (ID=1 of DB with ID=1 of TEST, then ID=2 of DB with ID=2 of TEST, and so on)

Output parameter OhDist contains the distances when hands from different persons are compared (ID=1 of DB with ID=2,3,4... of TEST, then ID=2 of DB with ID=1,3,4... of TEST, etc.)

```
[ShDist,OhDist] = DistNew('eucl',DB,TEST);

figure;
hold on;

% Histogram for same-person distances
histogram(ShDist, 20, 'Normalization', 'probability', ...
    'FaceColor', 'b', 'FaceAlpha', 0.5, 'EdgeColor', 'b');

% Histogram for different-person distances
histogram(OhDist, 50, 'Normalization', 'probability', ...
    'FaceColor', 'r', 'FaceAlpha', 0.5, 'EdgeColor', 'r');

xlabel('Distance');
ylabel('Probability');
title('Distance distributions: Same vs Different persons');
```

```
legend('Same Person','Different Persons');
grid on;
hold off;
print('histograms','-depsc');
```



With the help of the histograms you created earlier and shown in Figure = 100 (Genuine class) and figure = 101 (Impostor class), calculate and plot the probability of the FA error (= FAR) and FR error (= FRR) for various thresholds. You can also work directly with the distances in the vectors ShDist (Genuine) and OhDist (Impostor). Create a table with different thresholds and the number of errors of each type.

A Matlab command that you can use to calculate the number of FR errors at a specific threshold Th is: ***sum (ShDist> Th)****. Vary the value of Th and annotate both Th and the number of FR errors in a table.*

```
[ShDist,OhDist] = DistNew('eucl',DB,TEST);
th = linspace(0, 15, 200);

N_th = length(th);
FRR = zeros(1, N_th);
FAR = zeros(1, N_th);
FR_count = zeros(1, N_th);
FA_count = zeros(1, N_th);

for i = 1:length(th)
```

```
    % False Rejections (genuine > threshold)
    FR_count(i) = sum(ShDist > th(i));
    FRR(i) = FR_count(i) / numel(ShDist);

    % False Acceptances (impostor <= threshold)
    FA_count(i) = sum(OhDist <= th(i));
    FAR(i) = FA_count(i) / numel(OhDist);
end

% Create table for first few thresholds as an example
ErrorTable = table(th', FR_count', FA_count', FRR', FAR', ...
    'VariableNames', {'Threshold','FR_Errors','FA_Errors','FRR','FAR'});
disp(ErrorTable(1:200,:));  % show first 100 rows
```

```
    Threshold    FR_Errors    FA_Errors      FRR         FAR
    ---------    ---------    ---------    --------    ---------

           0        40             0        0.95238            0
    0.075377        40             0        0.95238            0
     0.15075        40             0        0.95238            0
     0.22613        40             0        0.95238            0
     0.30151        40             0        0.95238            0
     0.37688        40             0        0.95238            0
     0.45226        40             0        0.95238            0
     0.52764        40             0        0.95238            0
     0.60302        39             0        0.92857            0
     0.67839        39             0        0.92857            0
     0.75377        38             0        0.90476            0
     0.82915        38             0        0.90476            0
     0.90452        38             0        0.90476            0
      0.9799        37             0        0.88095            0
      1.0553        36             0        0.85714            0
      1.1307        36             0        0.85714            0
       1.206        36             0        0.85714            0
      1.2814        34             0        0.80952            0
      1.3568        33             0        0.78571            0
      1.4322        32             0         0.7619            0
      1.5075        30             0        0.71429            0
      1.5829        27             0        0.64286            0
      1.6583        26             0        0.61905            0
      1.7337        24             0        0.57143            0
       1.809        24             0        0.57143            0
      1.8844        23             0        0.54762            0
      1.9598        23             0        0.54762            0
      2.0352        21             0            0.5            0
      2.1106        21             2            0.5    0.0011614
      2.1859        19             4        0.45238    0.0023229
      2.2613        17             4        0.40476    0.0023229
      2.3367        17             4        0.40476    0.0023229
      2.4121        15             4        0.35714    0.0023229
      2.4874        13             4        0.30952    0.0023229
      2.5628        12             4        0.28571    0.0023229
      2.6382        11             5         0.2619    0.0029036
      2.7136        11             5         0.2619    0.0029036
```

2.7889	11	5	0.2619	0.0029036
2.8643	10	5	0.2381	0.0029036
2.9397	10	6	0.2381	0.0034843
3.0151	9	6	0.21429	0.0034843
3.0905	7	6	0.16667	0.0034843
3.1658	7	6	0.16667	0.0034843
3.2412	7	7	0.16667	0.004065
3.3166	6	8	0.14286	0.0046458
3.392	6	11	0.14286	0.0063879
3.4673	6	15	0.14286	0.0087108
3.5427	6	16	0.14286	0.0092915
3.6181	6	17	0.14286	0.0098722
3.6935	6	18	0.14286	0.010453
3.7688	6	19	0.14286	0.011034
3.8442	5	21	0.11905	0.012195
3.9196	5	24	0.11905	0.013937
3.995	5	27	0.11905	0.015679
4.0704	5	29	0.11905	0.016841
4.1457	5	30	0.11905	0.017422
4.2211	5	31	0.11905	0.018002
4.2965	5	33	0.11905	0.019164
4.3719	5	36	0.11905	0.020906
4.4472	4	41	0.095238	0.02381
4.5226	4	42	0.095238	0.02439
4.598	4	44	0.095238	0.025552
4.6734	4	46	0.095238	0.026713
4.7487	4	50	0.095238	0.029036
4.8241	4	56	0.095238	0.03252
4.8995	4	59	0.095238	0.034262
4.9749	4	63	0.095238	0.036585
5.0503	4	65	0.095238	0.037747
5.1256	4	70	0.095238	0.04065
5.201	4	78	0.095238	0.045296
5.2764	3	86	0.071429	0.049942
5.3518	3	90	0.071429	0.052265
5.4271	3	92	0.071429	0.053426
5.5025	3	97	0.071429	0.05633
5.5779	3	103	0.071429	0.059814
5.6533	3	107	0.071429	0.062137
5.7286	3	113	0.071429	0.065621
5.804	1	120	0.02381	0.069686
5.8794	1	126	0.02381	0.073171
5.9548	1	129	0.02381	0.074913
6.0302	1	132	0.02381	0.076655
6.1055	1	140	0.02381	0.081301
6.1809	1	143	0.02381	0.083043
6.2563	1	152	0.02381	0.088269
6.3317	1	155	0.02381	0.090012
6.407	1	165	0.02381	0.095819
6.4824	1	172	0.02381	0.099884
6.5578	1	177	0.02381	0.10279
6.6332	0	186	0	0.10801
6.7085	0	196	0	0.11382
6.7839	0	207	0	0.12021

6.8593	0	210	0	0.12195
6.9347	0	215	0	0.12485
7.0101	0	222	0	0.12892
7.0854	0	227	0	0.13182
7.1608	0	231	0	0.13415
7.2362	0	240	0	0.13937
7.3116	0	245	0	0.14228
7.3869	0	255	0	0.14808
7.4623	0	261	0	0.15157
7.5377	0	268	0	0.15563
7.6131	0	276	0	0.16028
7.6884	0	284	0	0.16492
7.7638	0	289	0	0.16783
7.8392	0	295	0	0.17131
7.9146	0	302	0	0.17538
7.9899	0	312	0	0.18118
8.0653	0	320	0	0.18583
8.1407	0	324	0	0.18815
8.2161	0	330	0	0.19164
8.2915	0	338	0	0.19628
8.3668	0	345	0	0.20035
8.4422	0	350	0	0.20325
8.5176	0	356	0	0.20674
8.593	0	363	0	0.2108
8.6683	0	368	0	0.2137
8.7437	0	377	0	0.21893
8.8191	0	390	0	0.22648
8.8945	0	396	0	0.22997
8.9698	0	404	0	0.23461
9.0452	0	411	0	0.23868
9.1206	0	423	0	0.24564
9.196	0	432	0	0.25087
9.2714	0	443	0	0.25726
9.3467	0	448	0	0.26016
9.4221	0	459	0	0.26655
9.4975	0	469	0	0.27236
9.5729	0	475	0	0.27584
9.6482	0	483	0	0.28049
9.7236	0	494	0	0.28688
9.799	0	506	0	0.29384
9.8744	0	510	0	0.29617
9.9497	0	520	0	0.30197
10.025	0	529	0	0.3072
10.101	0	534	0	0.3101
10.176	0	542	0	0.31475
10.251	0	550	0	0.3194
10.327	0	555	0	0.3223
10.402	0	564	0	0.32753
10.477	0	568	0	0.32985
10.553	0	579	0	0.33624
10.628	0	584	0	0.33914
10.704	0	591	0	0.34321
10.779	0	596	0	0.34611
10.854	0	605	0	0.35134

10.93	0	615	0	0.35714
11.005	0	626	0	0.36353
11.08	0	635	0	0.36876
11.156	0	642	0	0.37282
11.231	0	650	0	0.37747
11.307	0	654	0	0.37979
11.382	0	660	0	0.38328
11.457	0	665	0	0.38618
11.533	0	671	0	0.38966
11.608	0	680	0	0.39489
11.683	0	683	0	0.39663
11.759	0	687	0	0.39895
11.834	0	692	0	0.40186
11.91	0	697	0	0.40476
11.985	0	701	0	0.40708
12.06	0	707	0	0.41057
12.136	0	715	0	0.41521
12.211	0	720	0	0.41812
12.286	0	724	0	0.42044
12.362	0	734	0	0.42625
12.437	0	742	0	0.43089
12.513	0	758	0	0.44019
12.588	0	764	0	0.44367
12.663	0	771	0	0.44774
12.739	0	780	0	0.45296
12.814	0	787	0	0.45703
12.889	0	800	0	0.46458
12.965	0	808	0	0.46922
13.04	0	812	0	0.47154
13.116	0	817	0	0.47445
13.191	0	821	0	0.47677
13.266	0	832	0	0.48316
13.342	0	841	0	0.48839
13.417	0	847	0	0.49187
13.492	0	855	0	0.49652
13.568	0	862	0	0.50058
13.643	0	871	0	0.50581
13.719	0	881	0	0.51161
13.794	0	889	0	0.51626
13.869	0	892	0	0.518
13.945	0	899	0	0.52207
14.02	0	907	0	0.52671
14.095	0	913	0	0.5302
14.171	0	917	0	0.53252
14.246	0	929	0	0.53949
14.322	0	934	0	0.54239
14.397	0	942	0	0.54704
14.472	0	947	0	0.54994
14.548	0	950	0	0.55168
14.623	0	955	0	0.55459
14.698	0	963	0	0.55923
14.774	0	973	0	0.56504
14.849	0	982	0	0.57027
14.925	0	986	0	0.57259

```
         15          0            991               0       0.57549
```

Seperate Tables:

```
FRRTable = table(th', FR_count', FRR', ...
    'VariableNames', {'Threshold', 'FR_Errors', 'FRR'});
disp(FRRTable);
```

```
    Threshold    FR_Errors      FRR   
    _________    _________    ________

           0        40         0.95238
    0.075377        40         0.95238
     0.15075        40         0.95238
     0.22613        40         0.95238
     0.30151        40         0.95238
     0.37688        40         0.95238
     0.45226        40         0.95238
     0.52764        40         0.95238
     0.60302        39         0.92857
     0.67839        39         0.92857
     0.75377        38         0.90476
     0.82915        38         0.90476
     0.90452        38         0.90476
      0.9799        37         0.88095
      1.0553        36         0.85714
      1.1307        36         0.85714
       1.206        36         0.85714
      1.2814        34         0.80952
      1.3568        33         0.78571
      1.4322        32          0.7619
      1.5075        30         0.71429
      1.5829        27         0.64286
      1.6583        26         0.61905
      1.7337        24         0.57143
       1.809        24         0.57143
      1.8844        23         0.54762
      1.9598        23         0.54762
      2.0352        21             0.5
      2.1106        21             0.5
      2.1859        19         0.45238
      2.2613        17         0.40476
      2.3367        17         0.40476
      2.4121        15         0.35714
      2.4874        13         0.30952
      2.5628        12         0.28571
      2.6382        11          0.2619
      2.7136        11          0.2619
      2.7889        11          0.2619
      2.8643        10          0.2381
      2.9397        10          0.2381
      3.0151         9         0.21429
```

```
3.0905         7         0.16667
3.1658         7         0.16667
3.2412         7         0.16667
3.3166         6         0.14286
 3.392         6         0.14286
3.4673         6         0.14286
3.5427         6         0.14286
3.6181         6         0.14286
3.6935         6         0.14286
3.7688         6         0.14286
3.8442         5         0.11905
3.9196         5         0.11905
 3.995         5         0.11905
4.0704         5         0.11905
4.1457         5         0.11905
4.2211         5         0.11905
4.2965         5         0.11905
4.3719         5         0.11905
4.4472         4        0.095238
4.5226         4        0.095238
 4.598         4        0.095238
4.6734         4        0.095238
4.7487         4        0.095238
4.8241         4        0.095238
4.8995         4        0.095238
4.9749         4        0.095238
5.0503         4        0.095238
5.1256         4        0.095238
 5.201         4        0.095238
5.2764         3        0.071429
5.3518         3        0.071429
5.4271         3        0.071429
5.5025         3        0.071429
5.5779         3        0.071429
5.6533         3        0.071429
5.7286         3        0.071429
 5.804         1         0.02381
5.8794         1         0.02381
5.9548         1         0.02381
6.0302         1         0.02381
6.1055         1         0.02381
6.1809         1         0.02381
6.2563         1         0.02381
6.3317         1         0.02381
 6.407         1         0.02381
6.4824         1         0.02381
6.5578         1         0.02381
6.6332         0               0
6.7085         0               0
6.7839         0               0
6.8593         0               0
6.9347         0               0
7.0101         0               0
7.0854         0               0
```

7.1608	0	0
7.2362	0	0
7.3116	0	0
7.3869	0	0
7.4623	0	0
7.5377	0	0
7.6131	0	0
7.6884	0	0
7.7638	0	0
7.8392	0	0
7.9146	0	0
7.9899	0	0
8.0653	0	0
8.1407	0	0
8.2161	0	0
8.2915	0	0
8.3668	0	0
8.4422	0	0
8.5176	0	0
8.593	0	0
8.6683	0	0
8.7437	0	0
8.8191	0	0
8.8945	0	0
8.9698	0	0
9.0452	0	0
9.1206	0	0
9.196	0	0
9.2714	0	0
9.3467	0	0
9.4221	0	0
9.4975	0	0
9.5729	0	0
9.6482	0	0
9.7236	0	0
9.799	0	0
9.8744	0	0
9.9497	0	0
10.025	0	0
10.101	0	0
10.176	0	0
10.251	0	0
10.327	0	0
10.402	0	0
10.477	0	0
10.553	0	0
10.628	0	0
10.704	0	0
10.779	0	0
10.854	0	0
10.93	0	0
11.005	0	0
11.08	0	0
11.156	0	0

11.231	0	0
11.307	0	0
11.382	0	0
11.457	0	0
11.533	0	0
11.608	0	0
11.683	0	0
11.759	0	0
11.834	0	0
11.91	0	0
11.985	0	0
12.06	0	0
12.136	0	0
12.211	0	0
12.286	0	0
12.362	0	0
12.437	0	0
12.513	0	0
12.588	0	0
12.663	0	0
12.739	0	0
12.814	0	0
12.889	0	0
12.965	0	0
13.04	0	0
13.116	0	0
13.191	0	0
13.266	0	0
13.342	0	0
13.417	0	0
13.492	0	0
13.568	0	0
13.643	0	0
13.719	0	0
13.794	0	0
13.869	0	0
13.945	0	0
14.02	0	0
14.095	0	0
14.171	0	0
14.246	0	0
14.322	0	0
14.397	0	0
14.472	0	0
14.548	0	0
14.623	0	0
14.698	0	0
14.774	0	0
14.849	0	0
14.925	0	0
15	0	0

```
FARTable = table(th', FA_count', FAR', ...
    'VariableNames', {'Threshold', 'FA_Errors', 'FAR'});
disp(FARTable); % preview
```

```
    Threshold    FA_Errors       FAR
    _________    _________    _________

           0         0                0
    0.075377         0                0
     0.15075         0                0
     0.22613         0                0
     0.30151         0                0
     0.37688         0                0
     0.45226         0                0
     0.52764         0                0
     0.60302         0                0
     0.67839         0                0
     0.75377         0                0
     0.82915         0                0
     0.90452         0                0
      0.9799         0                0
      1.0553         0                0
      1.1307         0                0
       1.206         0                0
      1.2814         0                0
      1.3568         0                0
      1.4322         0                0
      1.5075         0                0
      1.5829         0                0
      1.6583         0                0
      1.7337         0                0
       1.809         0                0
      1.8844         0                0
      1.9598         0                0
      2.0352         0                0
      2.1106         2        0.0011614
      2.1859         4        0.0023229
      2.2613         4        0.0023229
      2.3367         4        0.0023229
      2.4121         4        0.0023229
      2.4874         4        0.0023229
      2.5628         4        0.0023229
      2.6382         5        0.0029036
      2.7136         5        0.0029036
      2.7889         5        0.0029036
      2.8643         5        0.0029036
      2.9397         6        0.0034843
      3.0151         6        0.0034843
      3.0905         6        0.0034843
      3.1658         6        0.0034843
      3.2412         7         0.004065
      3.3166         8        0.0046458
       3.392        11        0.0063879
```

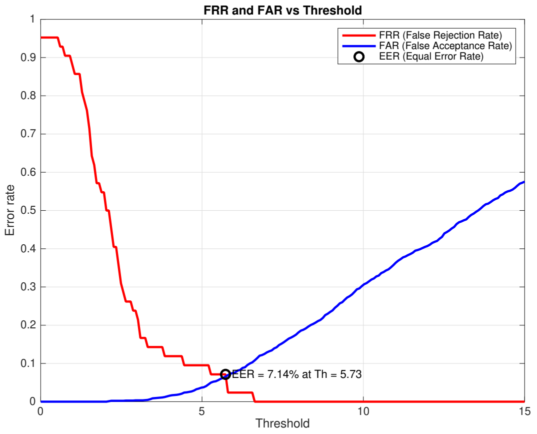
3.4673	15	0.0087108
3.5427	16	0.0092915
3.6181	17	0.0098722
3.6935	18	0.010453
3.7688	19	0.011034
3.8442	21	0.012195
3.9196	24	0.013937
3.995	27	0.015679
4.0704	29	0.016841
4.1457	30	0.017422
4.2211	31	0.018002
4.2965	33	0.019164
4.3719	36	0.020906
4.4472	41	0.02381
4.5226	42	0.02439
4.598	44	0.025552
4.6734	46	0.026713
4.7487	50	0.029036
4.8241	56	0.03252
4.8995	59	0.034262
4.9749	63	0.036585
5.0503	65	0.037747
5.1256	70	0.04065
5.201	78	0.045296
5.2764	86	0.049942
5.3518	90	0.052265
5.4271	92	0.053426
5.5025	97	0.05633
5.5779	103	0.059814
5.6533	107	0.062137
5.7286	113	0.065621
5.804	120	0.069686
5.8794	126	0.073171
5.9548	129	0.074913
6.0302	132	0.076655
6.1055	140	0.081301
6.1809	143	0.083043
6.2563	152	0.088269
6.3317	155	0.090012
6.407	165	0.095819
6.4824	172	0.099884
6.5578	177	0.10279
6.6332	186	0.10801
6.7085	196	0.11382
6.7839	207	0.12021
6.8593	210	0.12195
6.9347	215	0.12485
7.0101	222	0.12892
7.0854	227	0.13182
7.1608	231	0.13415
7.2362	240	0.13937
7.3116	245	0.14228
7.3869	255	0.14808
7.4623	261	0.15157

7.5377	268	0.15563
7.6131	276	0.16028
7.6884	284	0.16492
7.7638	289	0.16783
7.8392	295	0.17131
7.9146	302	0.17538
7.9899	312	0.18118
8.0653	320	0.18583
8.1407	324	0.18815
8.2161	330	0.19164
8.2915	338	0.19628
8.3668	345	0.20035
8.4422	350	0.20325
8.5176	356	0.20674
8.593	363	0.2108
8.6683	368	0.2137
8.7437	377	0.21893
8.8191	390	0.22648
8.8945	396	0.22997
8.9698	404	0.23461
9.0452	411	0.23868
9.1206	423	0.24564
9.196	432	0.25087
9.2714	443	0.25726
9.3467	448	0.26016
9.4221	459	0.26655
9.4975	469	0.27236
9.5729	475	0.27584
9.6482	483	0.28049
9.7236	494	0.28688
9.799	506	0.29384
9.8744	510	0.29617
9.9497	520	0.30197
10.025	529	0.3072
10.101	534	0.3101
10.176	542	0.31475
10.251	550	0.3194
10.327	555	0.3223
10.402	564	0.32753
10.477	568	0.32985
10.553	579	0.33624
10.628	584	0.33914
10.704	591	0.34321
10.779	596	0.34611
10.854	605	0.35134
10.93	615	0.35714
11.005	626	0.36353
11.08	635	0.36876
11.156	642	0.37282
11.231	650	0.37747
11.307	654	0.37979
11.382	660	0.38328
11.457	665	0.38618
11.533	671	0.38966

```
      11.608        680         0.39489
      11.683        683         0.39663
      11.759        687         0.39895
      11.834        692         0.40186
       11.91        697         0.40476
      11.985        701         0.40708
       12.06        707         0.41057
      12.136        715         0.41521
      12.211        720         0.41812
      12.286        724         0.42044
      12.362        734         0.42625
      12.437        742         0.43089
      12.513        758         0.44019
      12.588        764         0.44367
      12.663        771         0.44774
      12.739        780         0.45296
      12.814        787         0.45703
      12.889        800         0.46458
      12.965        808         0.46922
       13.04        812         0.47154
      13.116        817         0.47445
      13.191        821         0.47677
      13.266        832         0.48316
      13.342        841         0.48839
      13.417        847         0.49187
      13.492        855         0.49652
      13.568        862         0.50058
      13.643        871         0.50581
      13.719        881         0.51161
      13.794        889         0.51626
      13.869        892           0.518
      13.945        899         0.52207
       14.02        907         0.52671
      14.095        913          0.5302
      14.171        917         0.53252
      14.246        929         0.53949
      14.322        934         0.54239
      14.397        942         0.54704
      14.472        947         0.54994
      14.548        950         0.55168
      14.623        955         0.55459
      14.698        963         0.55923
      14.774        973         0.56504
      14.849        982         0.57027
      14.925        986         0.57259
          15        991         0.57549
```

```
% Plot FRR and FAR
figure;
plot(th, FRR, 'r-', 'LineWidth', 2); hold on;
plot(th, FAR, 'b-', 'LineWidth', 2);
xlabel('Threshold');
```

```
ylabel('Error rate');

% Add EER to the plot
plot(eer_threshold, eer_value, 'ko', 'MarkerSize', 8, 'LineWidth', 2);
text(eer_threshold, eer_value, sprintf('  EER = %.2f%% at Th = %.2f', eer_value*100, eer_threshold));

title('FRR and FAR vs Threshold');
legend('FRR (False Rejection Rate)', 'FAR (False Acceptance Rate)', 'EER (Equal Error Rate)');
grid on;
print('eer_plot','-depsc');
```



Choosing Threshholds: High Security vs. High Convenience

```
% High Security (FAR <= 1%)
sec_candidates = find(FAR <= 0.01);
[~, sec_best] = min(FRR(sec_candidates));
sec_idx = sec_candidates(sec_best);
fprintf('High Security → Th = %.4f, FAR = %.2f%%, FRR = %.2f%%\n', ...
    th(sec_idx), FAR(sec_idx)*100, FRR(sec_idx)*100);
```

```
High Security → Th = 3.3166, FAR = 0.46%, FRR = 14.29%
```

```
% High Convenience (FRR <= 1%)
conv_candidates = find(FRR <= 0.01);
[~, conv_best] = min(FAR(conv_candidates));
conv_idx = conv_candidates(conv_best);
fprintf('High Convenience → Th = %.4f, FAR = %.2f%%, FRR = %.2f%%\n', ...
    th(conv_idx), FAR(conv_idx)*100, FRR(conv_idx)*100);
```

```
High Convenience → Th = 6.6332, FAR = 10.80%, FRR = 0.00%
```

```
save('DB_final.mat', 'DB');
save('TEST_final.mat', 'TEST');
```