```
[x1,x2,x3,x4,v] = decoding
```

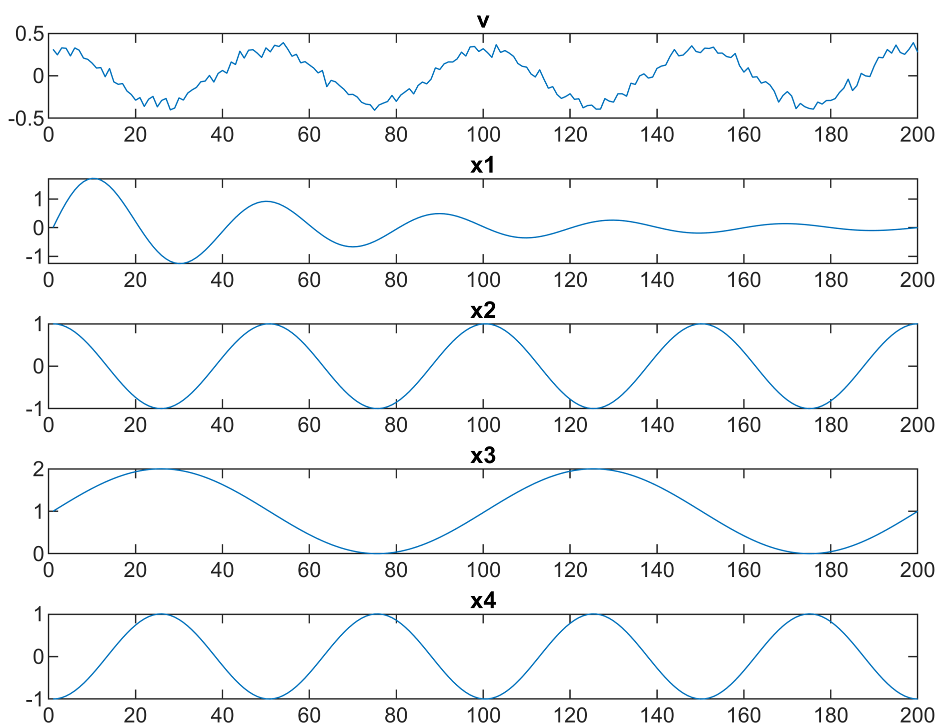
Problem 1: Decoding using inner products

```
x1 = 200×1
         0
    0.3095
    0.6017
    0.8700
    1.1084
    1.3120
    1.4769
    1.6000
    1.6798
    1.7156
      ⋮

x2 = 200×1
    1.0000
    0.9920
    0.9683
    0.9291
    0.8751
    0.8072
    0.7264
    0.6340
    0.5316
    0.4206
      ⋮

x3 = 200×1
    1.0000
    1.0631
    1.1260
    1.1883
    1.2499
    1.3105
    1.3699
    1.4278
    1.4840
    1.5382
      ⋮

x4 = 200×1
   -1.0000
   -0.9920
   -0.9683
   -0.9291
   -0.8751
   -0.8072
   -0.7264
   -0.6340
   -0.5316
   -0.4206
      ⋮

v = 200×1
    0.3117
    0.2474
    0.3290
    0.3241
    0.2344
    0.3286
    0.3016
    0.2095
    0.1935
```

```
0.1489
  ⋮
```

1a)

```
%a) Plotting vectors v, x1, x2, x3, x4
figure;
subplot(5,1,1);
plot(v);
title('v');

subplot(5,1,2);
plot(x1);
title('x1');

subplot(5,1,3);
plot(x2);
title('x2');

subplot(5,1,4);
plot(x3);
title('x3');

subplot(5,1,5);
plot(x4);
title('x4');
```



```
%Visually, it is obvious that the input signal x2 was used to generate v

%b) Calculating angle of v with each of the four signals xk.
```

1b)

```
X = [x1 x2 x3 x4];
theta = zeros(1,4);
for k = 1:4
    theta(k) = acosd(dot(v, X(:,k)) / (norm(v)*norm(X(:,k))));
end

[theta_min, k_min] = min(theta);
disp(theta);
```

```
   69.6662   10.9556   90.1134  169.0444
```

```
fprintf('Smallest angle: x^(%d) with %.4f degrees\n', k_min, theta_min);
```

```
Smallest angle: x^(2) with 10.9556 degrees
```

```
%Signal x2 makes the smallest angle with v, which confirms my conclusion
%that x2 was used to generate v.
```

```
[u,v,z] = multiaccess
```
Problem 2: multiaccess communication

```
u = 100×1
    -1
    -1
     1
     1
     1
    -1
     1
     1
    -1
    -1
     ⋮

v = 100×1
     1
     1
    -1
     1
    -1
     1
     1
     1
     1
     1
     ⋮

z = 500×1
   -1.8835
   -1.9373
    2.0075
    0.0352
    1.9303
   -1.8304
    0.0059
    0.1797
   -1.9736
   -1.9128
     ⋮
```

```
n = length(u);
m = length(z)/n;

%a) Calculate the angle between the code vectors u and v.

theta = acosd(dot(u,v)/(norm(u)*norm(v)));
fprintf('angle(u,v) = %.4f degrees\n', theta);
```

2a)

```
angle(u,v) = 92.2924 degrees
```

```
%How does this affect the decoding scheme? %Is it still possible to compute the
binary sequences b and c from z?
%Due to the angles being approximately orthogonal, each projection contains
%a small leakage, and with noise it makes the projection less exact,
```

```
%however since the angle is close to 90 degrees and the noise is small, the
%decoding scheme still works.

%Since the angle is near 90 degrees (92.2924) they are nearly orthogonal.

%b) Compute (b1,b2,...,b5) and (c1,c2,...,c5)   2b)
b_hat = zeros(m,1);
c_hat = zeros(m,1);

for k = 1:m
    zk = z((k-1) * n + (1:n));
    b_soft = dot(u, zk)/norm(u)^2;
    c_soft = dot(v, zk)/norm(v)^2;

    b_hat(k) = sign(b_soft);
    if b_hat(k) == 0
        b_hat(k) = 1;
    end

    c_hat(k) = sign(c_soft);
    if c_hat(k) == 0
        c_hat(k) = 1;
    end
end

disp('b (projection) =');
```

```
b (projection) =
```

```
disp(b_hat.')
```

```
     1    -1    -1     1    -1
```

```
disp('c (projection) =');
```

```
c (projection) =
```

```
disp(c_hat.')
```

```
    -1     1    -1    -1     1
```

3a) Regression line. Let $a, b \in \mathbb{R}^n$. $m_a = \text{avg}(a) = \frac{1^T a}{n}$, $m_b = \text{avg}(b) = \frac{1^T b}{n}$

$s_a = \text{std}(a) = \frac{1}{\sqrt{n}} \|a - m_a 1\|$, $s_b = \text{std}(b) = \frac{1}{\sqrt{n}} \|b - m_b 1\|$

We assume the vectors are not constant ($s_a \neq 0$ and $s_b \neq 0$) and write the correlation coefficient as $\rho = \frac{1}{n} \frac{(a - m_a 1)^T (b - m_b 1)}{s_a s_b}$

We considered the problem of fitting a straight line to the points (a_k, b_k) by minimizing $J = \frac{1}{n} \sum_{k=1}^{n} (c_1 + c_2 a_k - b_k)^2 = \frac{1}{n} \|c_1 1 + c_2 a - b\|^2$

Show that the optimal coefficients are $c_2 = \rho s_b / s_a$ and $c_1 = m_b - m_a c_2$. Show that for those values of c_1 and c_2, we have $J = (1 - \rho^2) s_b^2$.

Let $a_0 = a - m_a 1$ and $b_0 = b - m_b 1$

thus $s_a = \frac{1}{\sqrt{n}} \|a_0\|$ and $s_b = \frac{1}{\sqrt{n}} \|b_0\|$ and $\rho = \frac{1}{n} \cdot \frac{a_0^T b_0}{s_a s_b}$

$$J = \frac{1}{n} \|c_1 + c_2 a - b\|^2 = \frac{1}{n} \|c_1 + c_2 a_0 - b_0\|^2$$

$$\frac{\partial J}{\partial c_1} = \frac{2}{n} 1^T (c_1 1 + c_2 a_0 - b_0) = 2(c_1 + m_a c_2 - m_b) = 0$$

Thus $c_1 = m_b - m_a c_2$ ∎

Then $J(c_2) = \frac{1}{n} \|c_2 a_0 - b_0\|^2 = s_a^2 c_2^2 + s_b^2 - 2 c_2 \rho s_a s_b$

$J'(c_2) = 0$ so $2 s_a^2 c_2 - 2 \rho s_a s_b = 0$

$c_2 = \frac{\rho s_a s_b}{s_a^2} \Rightarrow$ thus $c_2 = \rho s_b / s_a$ ∎

Plug: $J(c_2) = \frac{1}{n} \|c_2 a_0 - b_0\|^2 = c_2^2 \frac{\|a_0\|^2}{n} + \frac{\|b_0\|^2}{n} - 2 c_2 \frac{a_0^T b_0}{n}$

$J(c_2) = s_a^2 c_2^2 + s_b^2 - 2 \rho s_a s_b c_2$

Thus $= s_a^2 (\rho \frac{s_b}{s_a})^2 + s_b^2 - 2 \rho s_a s_b (\rho \frac{s_b}{s_a})$

$= \rho^2 s_b^2 + s_b^2 - 2 \rho^2 s_b^2$

$= s_b^2 - \rho^2 s_b^2$

Thus $J = (1 - \rho^2) s_b^2$ ∎

3b) orthogonal distance regression

$\forall p \in (a_k, b_k)$, the vertical deviation from the straight line defined by $y = c_1 + c_2 x$ is given by $e_k = |c_1 + c_2 a_k - b_k|$

The orthogonal distance of (a_k, b_k) to the line: $d_k = \frac{|c_1 + c_2 a_k - b_k|}{\sqrt{1 + c_2^2}}$

We can find the straight line that minimizes the sum of the squared orthogonal distance $\sum J = \frac{1}{n} \sum_{k=1}^{n} d_k^2 = \frac{\|c_1 1 + c_2 a - b\|^2}{n(1 + c_2^2)}$

i) Show that the optimal value of c_1 is $c_1 = m_b - m_a c_2$ as for the least squares fit.

Let $r(c_1) = c_1 1 + c_2 a - b$

$$\frac{\partial}{\partial c_1} \|r(c_1)\|^2 = 2 \cdot 1^T r(c_1) = 2(nc_1 + c_2 1^T a - 1^T b) = 0$$

$$c_1 = \frac{1^T b}{n} - \frac{c_2 1^T a}{n}$$

$$c_1 = m_b - m_a c_2 \blacksquare$$

ii) $J = \frac{s_a^2 c_2^2 + s_b^2 - 2\rho s_a s_b c_2}{1 + c_2^2}$. Set $\frac{\partial J}{c_2} = 0$. Then $\rho c_2^2 + \left(\frac{s_a}{s_b} - \frac{s_b}{s_a}\right) c_2 - \rho = 0$

If $\rho = 0$ and $s_a = s_b$, any value of c_2 is optimal. If $\rho = 0$ and $s_a \neq s_b$ the quadratic eq. has a unique solution $c_2 = 0$. If $\rho \neq 0$, the quadratic eq. has a positive and a negative root. Show that the solution that minimizes J is the root c_2 with the same sign as ρ.

$$\rho c_2^2 + \left(\frac{s_a}{s_b} - \frac{s_b}{s_a}\right) c_2 - \rho = 0$$

If $\rho \neq 0$, the quadratic eq. has a positive and a negative root

Let r_1 and r_2 be roots. Thus $r_1 r_2 < 0$.

$$J'(c_2) = \frac{(2 s_a^2 c_2 - 2\rho s_a s_b)(1 + c_2^2) - (s_a^2 c_2^2 + s_b^2 - 2\rho s_a s_b c_2)(2c_2)}{(1 + c_2^2)^2}$$

$$J'(0) = \frac{-2\rho s_a s_b}{1} = -2\rho(s_a s_b)$$

If $\rho > 0$ then $J'(0) < 0$: The function is decreasing at 0. So positive c_2 is a minimum and negative c_2 is a max.

If $\rho < 0$ then $J'(0) > 0$: the function is increasing at 0. So, negative c_2 is minimum and positive is a maximum.

Thus, $\rho > 0$, a and b increase so the best fit slope should be positive

$\rho < 0$, a and b decrease so the best fit slope should be negative

Thus the solution that minimizes J is the root c_2 with the same sign as ρ $\blacksquare$

3b III)

```
a = [ ...
 8.96187818e-01
 7.00334212e-01
 8.81721369e-01
-5.51974562e-01
 1.15549181e+00
 1.00808409e+00
 2.06735762e+00
-4.21941514e-01
 3.89263477e+00
 4.09300943e-01
-5.05967311e-01
 3.18415295e+00
 4.09153094e+00
 5.99081715e-01
 3.00175441e+00
-8.92236884e-01
 3.47762254e+00
 2.31565957e+00
 4.84242891e+00
-1.14685189e+00
 9.15356281e-01
-4.37006646e-01
 3.60229313e+00
 3.01719934e+00
 4.45047474e+00
 3.68711482e+00
 1.67616823e-01
 3.88481663e+00
 1.30458035e+00
 2.94119198e+00
-4.10491162e-01
 4.27852249e+00
 3.49656316e+00
 4.69914047e+00
 3.09437776e+00
 2.62231653e+00
-6.10554928e-01
 4.15606384e+00
 1.22838450e+00
 3.63295477e-01
 1.97661400e+00
 3.87908624e+00
 1.54162804e+00
 1.59274164e+00
 1.44639584e+00
 2.94048129e+00
 5.73660875e-01
 2.63837661e+00
 4.20013278e-01
```

```
-7.75934138e-01
 3.82924050e+00
 6.52890325e-01
 2.39331709e-01
-1.11234409e+00
 1.38024714e+00
 1.36960933e+00
-4.39809329e-01
 1.80114112e+00
 7.45963345e-01
 3.07562547e+00
-5.92683561e-01
 3.61395575e+00
 3.67606427e+00
 4.27431438e+00
 1.04619577e+00
 4.44475860e+00
 3.58926088e+00
 3.44898344e-01
 5.03135688e-01
 1.74214618e+00
-5.27882686e-01
 2.78813886e+00
 2.97185217e-01
 1.52431020e+00
-1.11439522e+00
 1.59114450e+00
 3.66149808e+00
 3.91320845e+00
-5.26566932e-01
 3.93674275e+00
 5.79671364e-01
 2.55474236e+00
 1.77232716e+00
 4.08279010e-01
-1.18485408e-01
 3.50466914e+00
 4.54780895e+00
-4.30228369e-01
-9.23128528e-01
 3.84620658e+00
-5.35339049e-01
 2.62260250e+00
-7.00939752e-01
-2.40793560e-01
-1.04065355e+00
-9.91506907e-01
 4.93903510e+00
 3.04830768e+00
 4.76245349e-01
```

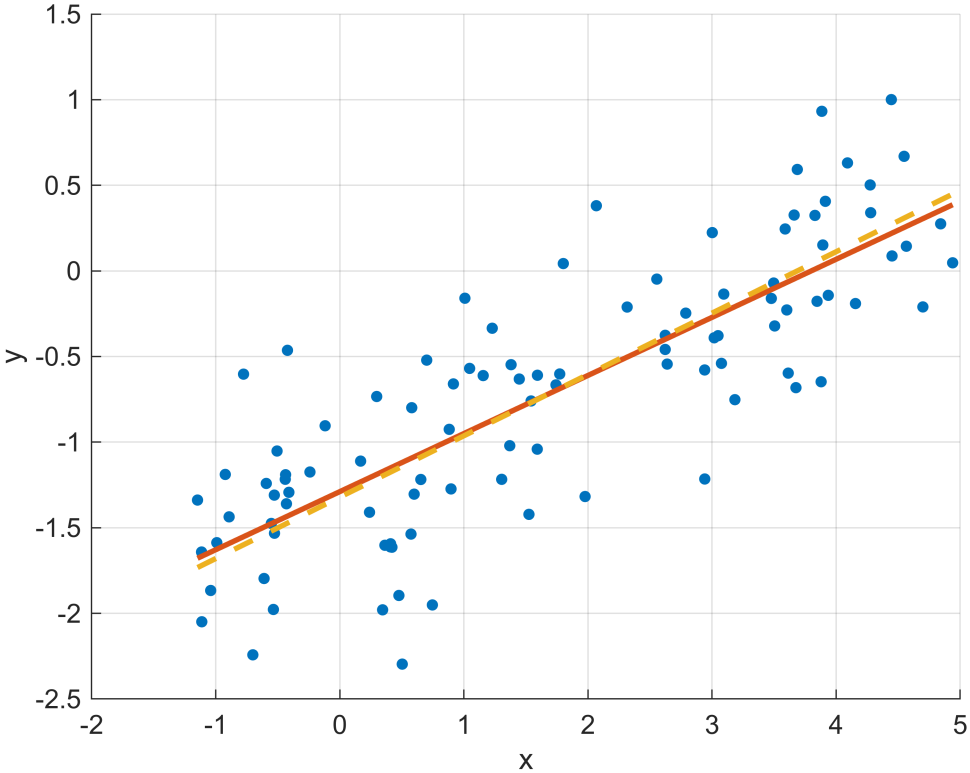
```
 4.56591874e+00 ];
b = [ ...
-1.27335109e+00
-5.21150668e-01
-9.25305115e-01
-1.47498575e+00
-6.11286856e-01
-1.59609056e-01
 3.80717831e-01
-4.63778539e-01
 1.51345870e-01
-1.59373196e+00
-1.05191572e+00
-7.52036807e-01
 6.30755630e-01
-1.30386428e+00
 2.23719499e-01
-1.43640681e+00
-1.60659892e-01
-2.11085069e-01
 2.74901558e-01
-1.33839156e+00
-6.60044243e-01
-1.19042322e+00
-2.28042842e-01
-3.90664199e-01
 8.73900750e-02
 5.92525550e-01
-1.11081232e+00
 9.32163700e-01
-1.21728830e+00
-1.21532981e+00
-1.29308721e+00
 3.39926107e-01
-7.07946097e-02
-2.10357953e-01
-1.35406484e-01
-4.58767989e-01
-1.79646696e+00
-1.90664088e-01
-3.34829323e-01
-1.60236373e+00
-1.31767549e+00
-6.47419085e-01
-7.59603865e-01
-6.09362761e-01
-6.31512117e-01
-5.78531171e-01
-1.53709451e+00
-5.44184333e-01
```

```
-1.61375667e+00
-6.02984285e-01
 3.24192448e-01
-1.21814386e+00
-1.40958778e+00
-2.04920884e+00
-5.47639543e-01
-1.02088879e+00
-1.21696448e+00
 4.29393420e-02
-1.95087963e+00
-5.39948017e-01
-1.24130785e+00
-5.96934417e-01
-6.81871324e-01
 5.01991798e-01
-5.69389371e-01
 1.00050225e+00
 2.44992259e-01
-1.97943394e+00
-2.29648571e+00
-6.65623227e-01
-1.30951652e+00
-2.46652220e-01
-7.33517983e-01
-1.42182443e+00
-1.64176082e+00
-1.04107182e+00
 3.26230021e-01
 4.06406975e-01
-1.53204187e+00
-1.42653002e-01
-7.98906514e-01
-4.78657927e-02
-6.02002881e-01
-1.61201110e+00
-9.04410097e-01
-3.21532418e-01
 6.69233494e-01
-1.36008289e+00
-1.18870878e+00
-1.77541296e-01
-1.97732743e+00
-3.76038189e-01
-2.24225080e+00
-1.17447048e+00
-1.86641627e+00
-1.58725492e+00
 4.73068313e-02
-3.78309640e-01
```

```
-1.89556472e+00
 1.44099079e-01 ];

n = length(a);
one = ones(n,1);

ma = mean(a);
mb = mean(b);

a0 = a - ma * one;
b0 = b- mb * one;

sa = norm(a0)/sqrt(n);
sb = norm(b0)/sqrt(n);

rh0 = (a0.'*b0)/(n*sa*sb);

c2_ls = rh0*sb/sa;
c1_ls = mb - ma*c2_ls;

A = rh0;
B = (sa/sb - sb/sa)
```

```
B = 1.9727
```

```
C = -rh0;

disc = B^2 - 4*A*C;
r1 = (-B + sqrt(disc))/(2*A);
r2 = (-B - sqrt(disc))/(2*A);

if sign(r1) == sign(rh0)
    c2_odr = r1;
else
    c2_odr = r2;
end
c1_odr = mb - ma*c2_odr;

x = linspace(builtin('min', a), builtin('max', a), 300).';
y_ls = c1_ls + c2_ls*x;
y_odr = c1_odr + c2_odr*x;

figure;
hold on;
grid on;
scatter(a, b, 18, 'filled');
plot(x, y_ls, 'LineWidth', 2);
plot(x, y_odr, '--', 'LineWidth', 2);
xlabel('x');
ylabel('y');
```


1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
y
-2
-1
0
1
2
3
4
5
x

Problem 4: The K-means algorithm

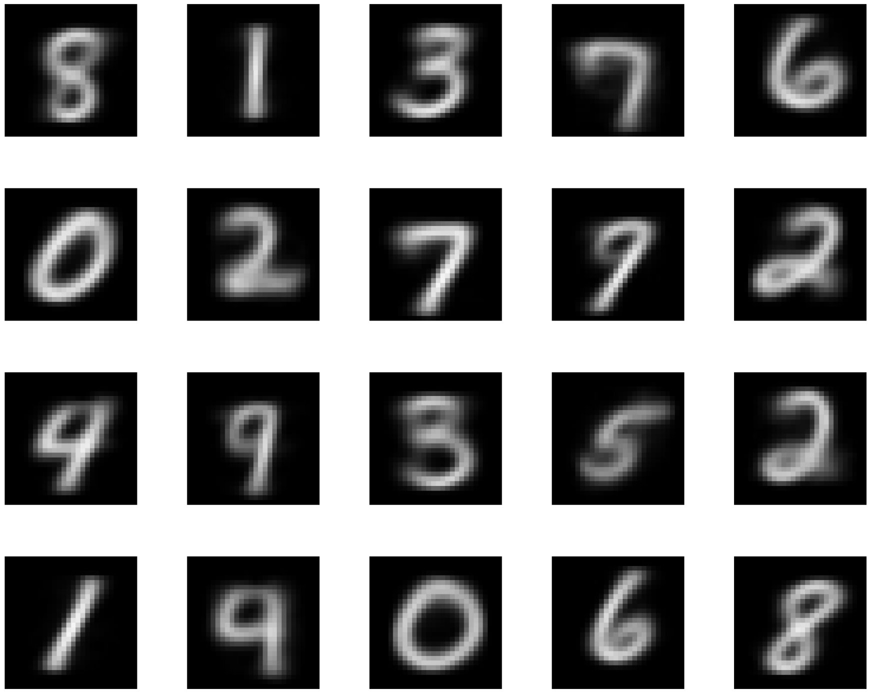
```
S = load('mnist_train.mat');
disp(S)
```

```
    digits: [784×60000 double]
    labels: [5 0 4 1 9 2 1 3 1 4 3 5 3 6 1 7 2 8 6 9 4 0 9 1 1 2 4 3 2 7 3 8 6 9 0 5 6 0 7 6 1 8 7 9 3 9 8 5 9 3 3 0
```

```
digits = digits(:,1:10000);

N = 10000;
k = 20;
group = randi(k, 1, N);

max = 50;
error_tol = 1e-5;
z = zeros(784,k);

for i = 1:max
    for j = 1:k
        idx = find(group == j);
        if ~isempty(idx)
            z(:,j) = mean(digits(:,idx),2);
        else
            z(:,j) = digits(:,randi(N));
        end
    end

    dists = zeros(k,N);
    for j = 1:k
        diff = digits - z(:,j);
        dists(j,:) = sum(diff.^2, 1);
    end
    [~, newGroup] = min(dists, [], 1);
    J = mean(min(dists, [], 1));
    if i > 1 && abs(J- Jprev) < error_tol * Jprev
        fprintf('Converged at i: %d\n', i);
        break;
    end
    Jprev = J;
    group = newGroup;
end
```

```
Converged at i: 42
```

```
figure;
for j = 1:k
    subplot(4, 5, j);
    imshow(reshape(z(:,j), 28, 28))
end
```



```
S = load('wikipedia_m.mat');
disp(S);
```

Problem 5: The K-means algorithm

```
      articles: {500×1 cell}
    dictionary: {4423×1 cell}
      tdmatrix: [4423×500 double]
```

```
if isfield(S, 'tdmatrix')
    X = S.tdmatrix;
    articles = S.articles;
    dictionary = S.dictionary;
else
    X = tdmatrix;
end

[d,N] = size(X);

k = 8;
i = 200;
error_tol = 1e-8;
numR = 5;

bestJ = inf;
best = struct();

for r = 1:numR
    fprintf("Restart %d/%d", r, numR);
    group = randi(k, 1, N);
    z = zeros(d, k);
    Jprev = inf;

    for j = 1:i
        for g = 1:k
            idx = find(group == g);
            if ~isempty(idx)
                z(:,g) = mean(X(:,idx),2);
            else
                z(:,g) = X(:,randi(N));
            end
        end

        dists = zeros(k, N);
        for g = 1:k
            D = X - z(:,g);
            dists(g,:) = sum(D.^2, 1);
        end
        [mind2, newGroup] = min(dists, [], 1);
        J = mean(mind2);

        if j > 1 && abs(J - Jprev) <= error_tol * Jprev
```

```
            fprintf("Converged in %d i. J = %.6g\n", j, J);
            break;
        end

        group = newGroup;
        Jprev = J;
        if j == i
            fprintf("Reached i. J = %.6g\n", J);
        end
    end

    if J < bestJ
        bestJ = J;
        best.z = z;
        best.group = group;
        best.J = J;
        best.dists = dists;
    end
end
```

```
Restart 1/5
Converged in 16 i. J = 0.00708804
Restart 2/5
Converged in 9 i. J = 0.00709056
Restart 3/5
Converged in 15 i. J = 0.00699051
Restart 4/5
Converged in 19 i. J = 0.00711807
Restart 5/5
Converged in 9 i. J = 0.00709408
```

```
fprintf("Best objective across restarts: J = %.6g", bestJ);
```

```
Best objective across restarts: J = 0.00699051
```

```
z = best.z;
group = best.group;

topTerms = cell(k,1);
closestArticle = cell(k,1);

for g = 1:k
    [~, ord] = sort(z(:,g), 'descend');
    topIdx = ord(1:min(5, numel(ord)));
    topTerms{g} = dictionary(topIdx);

    idx = find(group == g);
    if isempty(idx)
        closestArticle{g} = {'(empty cluster)'};
    else
        Dg = X(:,idx) - z(:,g);
```

```
        dd = sum(Dg.^2, 1);
        [~, loc] = sort(dd, 'ascend');
        pick = idx(loc(1:min(5, numel(loc))));
        closestArticle{g} = articles(pick);
    end
end

for g = 1:k
    fprintf(' Cluster %d\n', g);
    fprintf(' Top Terms: ');
    fprintf(' %s ', topTerms{g}{:});
    fprintf('\n Closest articles:\n');
    for j = 1:numel(closestArticle{g})
        fprintf('- %s\n', closestArticle{g}{j});
    end
end
```

```
 Cluster 1
 Top Terms:
 series  season  episode  film  television
 Closest articles:
- The_X-Files
- Charlie_Sheen
- Game_of_Thrones
- House_of_Cards_(U.S._TV_series)
- Supergirl_(U.S._TV_series)
 Cluster 2
 Top Terms:
 album  release  song  music  single
 Closest articles:
- David_Bowie
- Kanye_West
- Celine_Dion
- Ariana_Grande
- Kesha
 Cluster 3
 Top Terms:
 film  star  million  role  release
 Closest articles:
- Leonardo_DiCaprio
- Kate_Beckinsale
- Star_Wars:_The_Force_Awakens
- Star_Wars_Episode_I:_The_Phantom_Menace
- Maureen_O'Hara
 Cluster 4
 Top Terms:
 game  match  team  player  play
 Closest articles:
- Halo_5:_Guardians
- Fallout_4
- Call_of_Duty:_Black_Ops_III
- Overwatch_(video_game)
- Call_of_Duty:_Modern_Warfare_2
 Cluster 5
 Top Terms:
 match  championship  event  style  raw
 Closest articles:
- Wrestlemania_32
- Payback_(2016)
- Royal_Rumble_(2016)
```

```
- Night_of_Champions_(2015)
- Survivor_Series_(2015)
 Cluster 6
 Top Terms:
 season  win  game  team  player
 Closest articles:
- Kobe_Bryant
- Lamar_Odom
- Jose_Mourinho
- Johan_Cruyff
- Tom_Brady
 Cluster 7
 Top Terms:
 united  family  american  city  national
 Closest articles:
- Mahatma_Gandhi
- Sigmund_Freud
- Ben_Affleck
- Carly_Fiorina
- Frederick_Douglass
 Cluster 8
 Top Terms:
 fight  win  event  champion  fighter
 Closest articles:
- Floyd_Mayweather,_Jr.
- Kimbo_Slice
- Ronda_Rousey
- Jose_Aldo
- Joe_Frazier
```

```
S = load('tomography.mat');
disp(S);
```

Problem 6: Tomography

```
A: [576×784 double]
b: [576×1 double]
```

```
A = S.A;
b = S.b;

[m, n] = size(A);
K = 10;
error_tol = 1e-6;
x = zeros(n, 1);

rowNormSq = sum(A.^2, 2);
rowNormSq(rowNormSq == 0) = 1;

errs = zeros(K, 1);
prev_err = Inf;

for k = 1:K
    order = randperm(m);
    for idx = 1:m
        i = order(idx);
        ai = A(i, :).';
        x = x - ((ai.' * x - b(i))/rowNormSq(i)) * ai;
    end

    r = A*x - b;
    errs(k) = norm(r)/norm(b);
    fprintf("Cycle %d: relative error = %.6f\n", k, errs(k));

    if k > 1 && abs(errs(k) - prev_err) <= error_tol * builtin('max', prev_err, 1)
        break;
    end

    prev_err = errs(k);
end
```

```
Cycle 1: relative error = 0.140319
Cycle 2: relative error = 0.061345
Cycle 3: relative error = 0.034633
Cycle 4: relative error = 0.025148
Cycle 5: relative error = 0.018696
Cycle 6: relative error = 0.016192
Cycle 7: relative error = 0.014592
Cycle 8: relative error = 0.012685
Cycle 9: relative error = 0.011197
Cycle 10: relative error = 0.009468
```

```
figure;
```

```
imshow(reshape(x, 28, 28), []);
colormap gray;
axis image off;
```

```
figure;
plot(1:k, errs(1:k), '-o', 'LineWidth', 2);
grid on;
```