Q1. a

$$p(z_n = k \mid x_n) = \frac{p(x_n, z_n = k)}{p(x_n)} \quad (\because \text{Bayes Rule})$$

$$\Leftrightarrow p(z_n = k \mid x_n) = \frac{p(x \mid z_n = k)\, p(z_n = k)}{\sum_{j=1}^{K} p(x_n, z_n = j)}$$

$$\Leftrightarrow p(z_n = k \mid x_n) = \frac{\mathcal{N}(x_n \mid \mu_k, \Sigma_k)\, \pi_k}{\sum_{j=1}^{K} \mathcal{N}(x_n \mid \mu_j, \Sigma_j)\, \pi_j}$$

(Q.E.D)

Q 1. b

$$\underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{k} r_{nk} \log P(x_n, z_n = k \mid \theta)$$

$$\underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{k} r_{nk} \left\{ \log \mathcal{N}(x_n \mid \mu_k, \Sigma_k) + \log \pi_k \right\} = \alpha$$

1) find μ_k

$$\frac{\partial \alpha}{\partial \mu_k} = \sum_{n=1}^{N} \frac{r_{nk}}{\mathcal{N}(x_n \mid \mu_k, \Sigma_k)} \times \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \cdot \Sigma_k^{-1}(x_n - \mu_k)$$

$$= \sum_{n=1}^{N} r_{nk} \cdot \Sigma_k^{-1}(x_n - \mu_k)$$

$$= \Sigma_k^{-1} \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \qquad \Leftrightarrow \underline{\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} x_n} \;/\!/$$

2) find Σ_k

$$\frac{\partial \alpha}{\partial \Sigma_k} = \sum_{n=1}^{N} r_{nk} \left(\frac{1}{2} \Sigma_k^{-1} (x_n - \mu_k)(x_n - \mu_k)^T \Sigma_k^{-1} - \frac{1}{2} \Sigma_k^{-1} \right) = [0]$$

$$\Leftrightarrow \sum_{n=1}^{N} r_{nk} \left\{ \Sigma_k^{-1} (x_n - \mu_k)(x_n - \mu_k)^T \Sigma_k^{-1} - \Sigma_k^{-1} \right\} = [0]$$

$$\Leftrightarrow \sum_{n=1}^{N} r_{nk} (x_n - \mu_k)(x_n - \mu_k)^T - \Sigma_k \sum_{n=1}^{N} r_{nk} = [0] \Leftrightarrow \underline{\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (x_n - \mu_k)(x_n - \mu_k)^T} \;/\!/$$

3) find π_k, since $\sum_{k=1}^{k} \pi_k = 1$, instead of maximizing α, we will maximize

$$\alpha' = \alpha + \lambda \left(\sum_{k=1}^{k} \pi_k - 1 \right)$$

$$\frac{\partial \alpha'}{\partial \pi_k} = \sum_{n=1}^{N} \frac{r_{nk}}{\pi_k} + \lambda = 0 \Leftrightarrow \pi_k = -\frac{N_k}{\lambda}$$

Enforcing the constraint $\sum_{k=1}^{K} \pi_k = 1$, $\lambda = -N$

Therefore.

$$\underline{\pi_k = \frac{N_k}{N}} \;/\!/$$

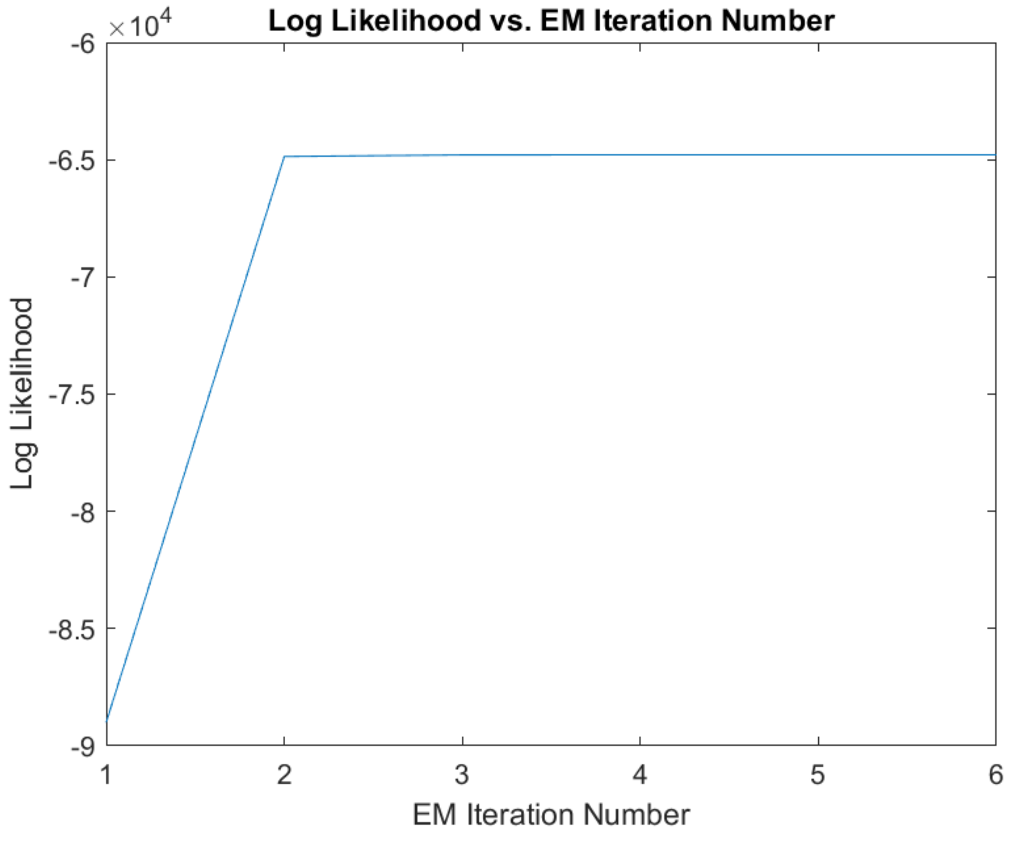
Q.2.A

Main Script

```
clear all; close all;
load("ps6_data.mat");
% Initialize the model parameters
K = 3;
mu = InitParams1.mu;
Sigma = repmat(InitParams1.Sigma, 1, 1, K);
pi = InitParams1.pi;
N = size(Spikes, 2);

% EM algorithm
maxIter = 100;
log_likelihood = zeros(maxIter, 1);

for iter = 1:maxIter
    % E-step
    log_gamma = zeros(K, N);
    for k = 1:K
        log_gamma(k, :) = log(pi(k)) + logmvnpdf(Spikes', mu(:, k)', Sigma(:, :, k));
    end
    logsumexp_gamma = logsumexp(log_gamma, 1);
    log_likelihood(iter) = sum(logsumexp_gamma);
    gamma = exp(log_gamma - logsumexp_gamma);

    % M-step
    Nk = sum(gamma, 2);
    for k = 1:K
        mu(:, k) = (Spikes * gamma(k, :)') / Nk(k);
        x_minus_mu = Spikes - mu(:, k);
        Sigma(:, :, k) = (x_minus_mu * diag(gamma(k, :)) * x_minus_mu') / Nk(k);
    end
    pi = Nk / N;

    % Check for convergence
    if iter > 1 && abs(log_likelihood(iter) - log_likelihood(iter-1)) < 1e-10
        log_likelihood = log_likelihood(1:iter);
        break;
    end
end

% Plot log likelihood
figure;
plot(1:length(log_likelihood), log_likelihood);
xlabel('EM Iteration Number');
ylabel('Log Likelihood');
title('Log Likelihood vs. EM Iteration Number');
```

```
function logpdf = logmvnpdf(x, mu, Sigma)
    d = length(mu);
    x_minus_mu = x - mu;
    invSigma = inv(Sigma);
    logpdf = -0.5 * (d * log(2 * pi) + log(det(Sigma)) + sum((x_minus_mu * invSigma)
.* x_minus_mu, 2));
end
```

Result



Q.2.B

Main script

```
% Store estimates
mu_k = mu;
sigma_k = Sigma;
pi_k = pi;

% Display pi_k
disp('pi_k:');
disp(pi_k);
```

Result

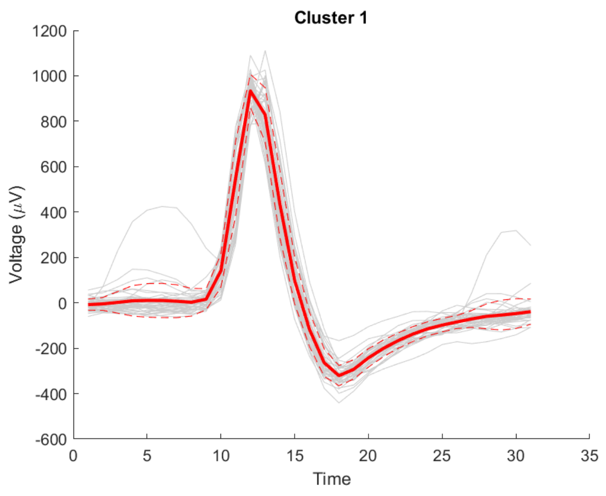
```
pi_k:
    0.0761
    0.8333
    0.0906
```

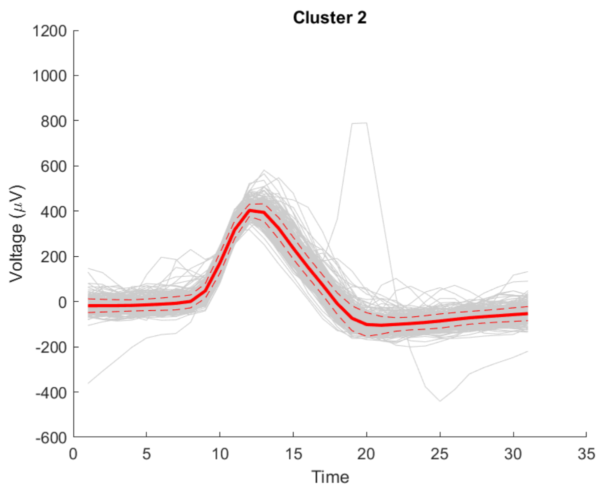
Q.2.C

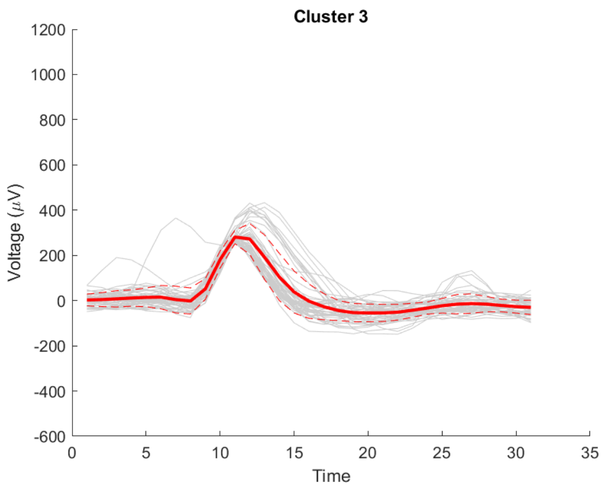
Main Script

```
% Plot voltage vs time for each cluster
for k = 1:K
    figure;
    hold on;

    % Plot waveform snippets assigned to the kth neuron
    [~, cluster_assignments] = max(gamma, [], 1);
    plot(Spikes(:, cluster_assignments == k), 'Color', [0.8 0.8 0.8]);

    % Plot cluster center and standard deviation bounds
    plot(mu_k(:, k), 'r', 'LineWidth', 2);
    plot(mu_k(:, k) + sqrt(diag(sigma_k(:, :, k))), 'r--');
    plot(mu_k(:, k) - sqrt(diag(sigma_k(:, :, k))), 'r--');

    xlabel('Time');
    ylabel('Voltage (\muV)');
    title(['Cluster ' num2str(k)]);
    ylim([-600 1200])
    hold off;
end
```

Result







Q.3

Main Scirpt

```
% Initialize the model parameters
K = 3;
mu = InitParams2.mu;
Sigma = repmat(InitParams2.Sigma, 1, 1, K);
pi = InitParams2.pi;
N = size(Spikes, 2);

% the same script below as Question 2
```

Results

Matlab shows the warning: **"Matrix is close to singular or badly scaled. Results may be inaccurate."**

This warning occurs when the covariance matrix Sigma becomes ill-conditioned or close to singular during the EM algorithm. It means that the matrix is either not full rank or has very large or very small eigenvalues, which can lead to numerical instability when computing the matrix inverse or determinant.

To address when the sigma becomes ill-conditioned, I add a script in the M-step

```
    % M-step
    Nk = sum(gamma, 2);
    for k = 1:K
        mu(:, k) = (Spikes * gamma(k, :)') / Nk(k);
        x_minus_mu = Spikes - mu(:, k);
        Sigma(:, :, k) = (x_minus_mu * diag(gamma(k, :)) * x_minus_mu') / Nk(k);

        % Check if the covariance matrix is well-conditioned and invertible
        cond_number = cond(Sigma(:, :, k));
        determinant = det(Sigma(:, :, k));
        if cond_number >= 1e10 || determinant <= 1e-10
            fprintf('Warning: Ill-conditioned covariance matrix at iteration %d,
cluster %d\n', iter, k);
            fprintf('Condition number: %g, determinant: %g\n', cond_number,
determinant);
        end
    end
```

and output was that

```
Warning: Ill-conditioned covariance matrix at iteration 2, cluster 1
Condition number: 8.77433e+18, determinant: -1.4627e-216
```

Which means in the second iteration, the updated sigma in cluster1 becomes ill-conditioned. I think it is caused by the degenerate solutions where one or more clusters have an extremely small covariance matrix, effectively modeling a single point or a very tight group of points.