Task 1)

a)

```matlab
%% mean %
for i=1:M
    xmix = xmix+w(i)*x(:,i);
end

%% covariance

% first term in covariance of gaussian mixture
Pmix1 = zeros(n,n);
for i=1:M
    Pmix1 = Pmix1+w(i)*P(:,:,i);
end

% second term, spread of the innovations term
Pmix2 = zeros(n,n);
for i=1:M
    Pmix2 = Pmix2+w(i)*x(:,i)*x(:,i)';
end
Pmix2 = Pmix2-xmix*xmix';

Pmix = Pmix1+Pmix2;

end
```

b)

<u>Gaussian mixture:</u>

$$f(x) = \sum_{i=1}^{M} w^i \mathcal{N}(x; \mu^i, P^i) \quad , \sum w^i = 1 , \; w^i \geq 0 \; \forall i$$

$$\bar{\mu} = \sum_{i=1}^{M} w^i \mu^i \quad , \quad \bar{P} = \sum w^i P^i + \tilde{P}$$

$$\tilde{P} = \sum w^i \mu^i (\mu^i)^T - \bar{\mu}\bar{\mu}^T$$: spread of the innovations term: how much individual expectations differ

"Sufficiently similar" is the key characteristic to look for here

i) 1 & 2 : same shape, as well as slightly larger variance
$\bar{\mu} = 1$, $\bar{\sigma}^2 = 2$

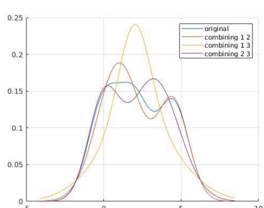
ii) 1 & 2 : (1,3) is narrow ⇒ bad, between (1,2) & (2,3), (1,2) – better μ & almost same variance
$\bar{\mu}$ 1.6 , $\bar{\sigma}^2 = 1.64$

iii) 2, 3 – best shape to approximate,
$\bar{\mu} = 3.81$, $\bar{\sigma}^2 = 1.95$

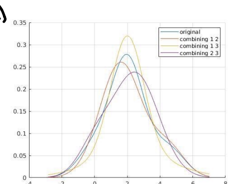
iv) 2, 3 – best shape to approximate
$\bar{\mu} = 3.81$, $\bar{\sigma}^2 = 1.95$

<u>Visualization</u>

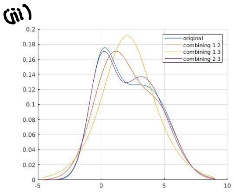
(i)



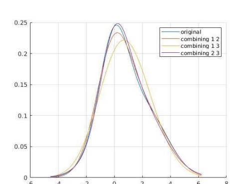
(ii)



(iii)



(iv)



Task 2)

a) $p(z_k | z_{1:k-1}) = \sum_{s_k} \int \underbrace{p(z_k | x_k, s_k)}_{p(z_k | x_k, s_k, z_{1:k-1})} \, p(x_k | s_k, z_{1:k-1}) \Pr(s_k | z_{1:k-1}) \, dx_k$

measurement at timestep k is independent of z_{k-1}, when x_k is given, so equality holds.

$$= \sum_{s_k} \int \underbrace{p(z_k | x_k, s_k, z_{1:k-1}) \, p(x_k | s_k, z_{1:k-1})}_{\text{bayes}} \Pr(s_k | z_{1:k-1}) \, dx_k$$

$$= \sum_{s_k} \underbrace{\int p(z_k, x_k | s_k, z_{1:k-1}) \, dx_k}_{\Lambda_k^{(s_k)}} \Pr(s_k | z_{1:k-1})$$

$$= \underline{\underline{\sum_{s_k} \Lambda_k^{(s_k)} \Pr(s_k | z_{1:k-1})}}$$

b)

$$p(z_k | z_{1:k-1}) \overset{\text{total prob-theorem}}{=} \int p(z_k, x_k | z_{1:k-1}) \, dx_k$$

$$= \int p(z_k | x_k) \, p(x_k | z_{1:k-1}) \, dx_k$$

$$= \int p(z_k | x_k) \sum_{i=1}^{N} w_k^i \, \delta(x_k - x_k^i) \, dx_k$$

$$= \sum_{i=1}^{N} w_k^i \int p(z_k | x_k) \, \delta(x_k - x_k^i) \, dx_k$$

$$= \underline{\underline{\sum_{i=1}^{N} w_k^i \, p(z_k | x_k^i)}}$$

Task 3)

step 1

smixprobs:

$$6.26: \ \mu_{s_{k-1}|s_k} = \Pr\{s_{k-1} | s_k, z_{1:k-1}\}$$

$$= \frac{\overbrace{\Pr\{s_k | s_{k-1}, z_{1:k-1}\} \Pr\{s_{k-1} | z_{1:k-1}\}}^{\text{joint distribution}}}{\underbrace{\Pr\{s_k | z_{1:k-1}\}}_{\text{spread probs}}}$$

$\Pr\{s_{k-1}, s_k\} = \Pr\{s_{k-1} | s_k\} \Pr\{s_k\}$

Step 4) loglikelihood: $p(z_k | z_{1:k-1}) = \sum_{s_k} \Lambda_k^{s_k} \Pr\{s_k | z_{1:k-1}\}$

node likelihood

spredprob

Task 3)

a)

```
%% NOTE: s_k along row(i) axis and s_k-1 along column(j) axis

%% Joint probability for this model and next
% numerator of (eq 6.26) (M x M)
spsjointprobs = zeros(obj.M,obj.M);
for i=1:obj.M
    for j=1:obj.M
        spsjointprobs(i,j) = obj.PI(i,j)*sprobs(j);
    end
end

%% marginal probability for next model
% denominator of (eq 6.26), normalization constant (M x 1)
spredprobs = obj.PI*sprobs;

%% conditionional probability for model at this time step on the next.
% (eq 6.26) (M x M)
smixprobs = zeros(obj.M,obj.M);
for i=1:obj.M
    for j=1:obj.M
        smixprobs(i,j) = spsjointprobs(i,j)/spredprobs(i);
    end
end
```

b)

```
% allocate
xmix = zeros(size(x));
Pmix = zeros(size(P));

% mix for each mode
for i=1:obj.M
    [xmix(:,i), Pmix(:,:,i)] = reduceGaussMix(smixprobs(i,:), x(:,i), P(:,:,i))
end
```

c)

```
%% mode matched prediction
for i=1:obj.M
    [xpred(:,i), Ppred(:,:,i)] = obj.modeFilters{i}.predict(x(:,i),P(:,:,i),Ts)
end
```

d)

```
% step 1
[spredsprob,smixprobs] = obj.mixProbabilities(sprobs);

% step 2
[xmix,Pmix] = obj.mixStates(smixprobs,x,P);

% prediction part of step 3
[xpred,Ppred] = obj.modeMatchedPrediction(xmix,Pmix,Ts);
```

e)

```
% mode matched update and likelihood
for i=1:obj.M
    [xupd(:,i),Pupd(:,:,i)] = obj.modeFilters{i}.update(z,x(:,i),P(:,:,i));
    logLambdas(i) = obj.modeFilters{i}.loglikelihood(z,x(:,i),P(:,:,i));
end
```

f)

```
[spredsprob,smixprobs] = obj.mixProbabilities(sprobs);

loglikelihood = logSumExp(logLambdas+log(spredprobs));  % Denomenator of eq 6.26
supdprobs = exp(logLambdas+log(spredsprob))/exp(loglikelihood);  % eq 6.32
```

g)

```
% update part of step 3
[xupd, Pupd, logLambdas] = obj.modeMatchedUpdate(z,x,P);

% step 4
[supdprobs, loglikelihood] = obj.updateProbabilities(logLambdas,sprobs);
```

h) This was given:

```
[xest, Pest] = reduceGaussMix(sprobs, x, P);
```

i) Took a look ☺