```
function [t_out, x_out, P_out, mu_out] = static_mm_filter(F, G, Gamma, H, Q,
R, u, z, xhat0, P0, varargin)
%STATIC_MM_FILTER Linear Multiple Model filter with the assumption that true
model
% parameters are constant in time

lower_bound = 0;

%--- loading any optional arguments
while ~isempty(varargin)
    switch lower(varargin{1}) % switch to lowercase only
        case 'lowerbound'
              lower_bound = varargin{2};
        otherwise
              error(['Unexpected option: ' varargin{1}])
    end
    varargin(1:2) = [];
end

nx = length(xhat0);
nz = size(H, 1);
num_meas = size(z, 1)/nz;
num_models = size(F, 3);


% Initialize output
x_out = zeros(nx, num_models+1, num_meas+1); % Save all estimates plus the
fused
P_out = zeros(nx, nx, num_models+1, num_meas+1);
t_out = 0:num_meas;
mu_out = zeros(num_meas+1, num_models);

% Initialize filters
mu = ones(1, num_models)/num_models; % Start with equal probability
nu = zeros(nz, num_models);
S = zeros(nz, nz, num_models);

x_out(:, :, 1) = repmat(xhat0, 1, num_models+1); % Save all the initial
guesses
P_out(:, :, :, 1) = repmat(P0, 1, 1, num_models+1); % Save all the initial
covariances
mu_out(1, :) = mu;

xhat = repmat(xhat0, 1, num_models);
P = repmat(P0, 1, 1, num_models);

for k=1:num_meas
    % Run all the filters one step
    for j=1:num_models
    [~, ~, xhat(:, j), P(:, :, j), nu(:, j), S(:, :, j)] = kf_step(F(:, :,
j), G(:, :, j), Gamma(:, :, j), H(:, :, j), Q(:, :, j), R(:, :, j), u(k, :),
z(k, :), xhat(:, j), P(:, :, j));
```

```
    end

    % Save the estimates
    x_out(:, 1:num_models, k+1) = xhat;
    P_out(:, :, 1:num_models, k+1) = P;

    % Update the weights
    for i=1:num_models
        mu(i) = mu(i) * mvnpdf(nu(:, i), zeros(size(nu(:, i))), S(:, :, i));
% There's a Cholesky way to get this that is faster
        % ad hoc lower bound
        if mu(i) < lower_bound
            mu(i) = lower_bound;
        end
    end
    mu = mu / sum(mu); % Normalize the weights
    mu_out(k+1, :) = mu;

    % Calculate the fused estimate
    x_fused = xhat*mu.';
    P_fused = zeros(nx, nx);
    for m=1:num_models % Sure there's a way to vectorize this a little
smarter
        P_fused = P_fused + mu(m) * (P(:, :, m) + (xhat(:, m) -
x_fused)*(xhat(:, m) - x_fused).');
    end

    % Save the fused estimates
    x_out(:, end, k+1) = x_fused;
    P_out(:, :, end, k+1) = P_fused;

end
```

Published with MATLAB® R2025a