Task 1)

- At most 1 boat
- $r_k \in (0,1)$, timestep k, boat exists
- stay P_S, leave $1-P_S$
- enter P_E, not enter $1-P_E$
- P_D boat present - radar.
- P_{FA} false alarm

Bayesian filter

a) $P(x_k) = r_k$, x_k : boat exists at time k

$$\underbrace{P(x_{k+1} \mid x_k)}_{\text{predicted state estimate}} = \underline{\underline{r_k P_S + (1-r_k) P_E}} \quad , \quad P(x_{k+1} \mid \bar{x}_k) = (1-r_k) P_S + r_k P_E$$

Task 1b)

a) $\hat{r}_{k+1} = P(\hat{R}_{k+1|k} = 1 \mid z=0) + P(\hat{R}_{k+1|k} = 1 \mid z=1)$

$$P(\hat{R}_{k+1|k} = 1 \mid z=0) = \frac{P(z=0 \mid R_{k+1|k} = 1)\, P(\hat{R}_{k+1|k} = 1)}{P(z=0)}$$

z=1: får måling
x_k: båt er i region

$$P(z=0) = P(z=0 \mid x_k = 1) + P(z=0 \mid x_k = 0)$$
$$= \underline{\underline{(1-P_D)(1-P_{FA}) + (1-P_{FA})}}$$

Task 2)

a) $\hat{x}_1 = \begin{bmatrix} \hat{p}_1 \\ \hat{u}_1 \end{bmatrix} = \begin{bmatrix} K_{p_1} & K_{p_0} \\ K_{u_1} & K_{u_0} \end{bmatrix} \begin{bmatrix} z_1 \\ z_0 \end{bmatrix}$, $\hat{p}_1 = \begin{bmatrix} \hat{p}_{x1} \\ \hat{p}_{y1} \end{bmatrix}$, $\hat{u}_1 = \begin{bmatrix} \hat{u}_{x1} \\ \hat{u}_{y1} \end{bmatrix}$

$$z_k = \begin{bmatrix} z_{xk} \\ z_{yk} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{xk} \\ p_{yk} \\ u_{xk} \\ \vdots \end{bmatrix} + \begin{bmatrix} w_{xk} \\ w_{yk} \end{bmatrix} = P_k + w_k$$

$H \qquad \begin{bmatrix} u_{x_k} \\ u_{y_k} \end{bmatrix}$

$z_k = I\, p_k + \omega_k \qquad p_k = (z_k - \omega_k) \;,$

$$\begin{cases} z_1 = p_1 + \omega_1 & , \; \omega_k \sim \mathcal{N}(0, R) \\ z_0 = p_0 + \omega_0 \end{cases}$$

<u>cv model:</u>

$x_1 = F x_0 + v_0$

$x_1 = F x_0 + v_0$

$$\begin{bmatrix} p_{x_1} \\ p_{y_1} \\ u_{x_1} \\ u_{y_1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{F} \begin{bmatrix} p_{x_0} \\ p_{y_0} \\ u_{x_0} \\ u_{y_0} \end{bmatrix} + v_0$$

$x_0 = F^{-1}(x_1 - v_0) \qquad u_1 = u_0 + v_0$

$z_0 = H x_0 + \omega_0$

$= \underline{\underline{H F^{-1}(x_1 - v_0) + \omega_0}} = \underline{\underline{p_1 - T u_1 - H F^{-1} v_0 + \omega_0}}$

$\underline{\underline{z_1 = p_1 + \omega_1}}$

b) $E[\hat{x}_1] = x_1$

$$E\left[K \begin{bmatrix} z_1 \\ z_0 \end{bmatrix}\right] = \begin{bmatrix} p_1 \\ u_1 \end{bmatrix}$$

$$E\left[\begin{bmatrix} K_{p_1} & K_{p_0} \\ K_{u_1} & K_{u_0} \end{bmatrix} \begin{bmatrix} z_1 \\ z_0 \end{bmatrix}\right] = \begin{bmatrix} p_1 \\ u_1 \end{bmatrix}$$

$$\underset{\substack{\uparrow \\ \text{constant} \\ \text{matrise}}}{K} \cdot E\begin{bmatrix} z_1 \\ z_0 \end{bmatrix} = K \begin{bmatrix} p_1 \\ p_1 - T u_1 \end{bmatrix}$$

$$= \begin{bmatrix} K_{p_1} p_1 + K_{p_0}(p_1 - T u_1) \\ K_{u_1} p_1 + K_{u_0}(p_1 - T u_1) \end{bmatrix}$$

$$= \begin{bmatrix} p_1 \\ u_1 \end{bmatrix} \quad \text{när} \quad \boxed{\begin{array}{l} K_{p_1} = I_2 \\ K_{p_0} = O_2 \\ K_{u_1} = \frac{I_2}{T} \\ K_{u_0} = -\frac{I_2}{T} \end{array}}$$

c) $$\mathrm{Cov}(\hat{x}) = K\,\mathrm{Cov}(z)K^T$$

$$\mathrm{Cov}\begin{bmatrix} p_1 + w_1 \\ p_1 - Tu_1 - HF^{-1}v_0 \\ w_0 \end{bmatrix} = \mathrm{Cov}(z) = \mathrm{Cov}\begin{pmatrix} z_1 \\ z_0 \end{pmatrix} = \begin{bmatrix} R & 0 \\ 0 & R + (HF^{-1})R(HF^{-1})^T \end{bmatrix}$$

$$\mathrm{Cov}(\hat{x}) = K \begin{bmatrix} R & 0 \\ 0 & R + (HF^{-1})R(HF^{-1})^T \end{bmatrix} K^T$$

d) $$x = f(\hat{x}, w, v)$$

x = lin(random var), $\hat{x}$ given as constant

$$\hat{x}_1 = K\begin{bmatrix} z_1 \\ z_0 \end{bmatrix} = \begin{bmatrix} I_2 & O_2 \\ \frac{I_2}{T} & -\frac{I_2}{T} \end{bmatrix}\begin{bmatrix} z_1 \\ z_0 \end{bmatrix}$$

$$= \begin{bmatrix} z_1 \\ \frac{z_1}{T} - \frac{z_0}{T} \end{bmatrix}$$

$$= \begin{bmatrix} p_1 + w_1 \\ \cancel{\frac{p_1}{T}} + \frac{w_1}{T} - \cancel{\frac{p_1}{T}} + u_1 + \frac{HF^{-1}v_0}{T} - \frac{w_0}{T} \end{bmatrix}$$

$$= \begin{bmatrix} p_1 + w_1 \\ u_1 + \frac{HF^{-1}v_0}{T} - \frac{w_0}{T} \end{bmatrix} + \frac{w_1}{T}$$

$$x = x + \begin{bmatrix} w_1 \\ \frac{HF^{-1}v_0}{T} - \frac{w_0}{T} \end{bmatrix} + \frac{w_1}{T}$$

$$x = \hat{x} - \begin{bmatrix} w_1 \\ \frac{w_1}{T} + \frac{HF^{-1}v_0}{T} - \frac{w_0}{T} \end{bmatrix} = \hat{x} - Bw_1 + Cw_0 + Dv_0$$

$$\underline{E[x_1] = \hat{x}_1}$$

$$Cov(x_1) = B\underbrace{Cov(w)}_{R}B^T + C\underbrace{Cov(w_0)}_{R}C^T + D\underbrace{Cov(v_0)}_{Q}D^T$$

$$= \underline{\underline{BRB^T + CRC^T + DQD^T}}$$

$$\underline{\underline{x \sim \mathcal{N}(\hat{x}; cov(x_1))}}$$

e) theoretically optimal

Task 3) a) → e)

```
function [xp, Pp] = predict(obj, x, P, Ts)
    % returns the predicted mean and covariance for a time step Ts
    Fk = obj.F(x, Ts);

    xp = obj.f(x, Ts);
    Pp = Fk*P*Fk'+obj.Q(x,Ts);
end

function [vk, Sk] = innovation(obj, z, x, P)
    % returns the innovation and innovation covariance
    Hk = obj.H(x);
    zp = obj.h(x);
    vk = z-zp;
    Sk = Hk*P*Hk'+obj.R(x,z);
end
```

```
function [xupd, Pupd] = update(obj, z, x, P)
    % returns the mean and covariance after conditioning on the
    % measurement
    [vk, Sk] = obj.innovation(z, x, P);
    Hk = obj.H(x);
    I = eye(size(P));

    Wk = P*Hk'inv(Sk);

    xupd = x+Wk*vk;
    Pupd = (I-Wk*Hk)*P;
end

function NIS = NIS(obj, z, x, P)
    % returns the normalized innovation squared
    [vk, Sk] = obj.innovation(z, x, P);

    NIS = vk'*inv(Sk)*vk;
end

function ll = loglikelihood(obj, z, x, P)
    % returns the logarithm of the marginal mesurement distribution
    [vk, Sk] = obj.innovation(z, x, P);
    NIS = obj.NIS(z, x, P);

    ll = -0.5*NIS;
end
```

Task 4)

a) → f)

```
function model = discreteCVmodel(q, r)
    % returns a structure that implements a discrete time CV model with
    % continuous time accelleration covariance q and positional
    % measurement with noise with covariance r, both in two dimensions.

    % discrete prediction function
    model.f = @(x, Ts) [1 0 Ts 0;
                        0 1 0 Ts;
                        0 0 1 0;
                        0 0 0 1]*x;

    % jacobian of prediction function
    model.F = @(x, Ts) [1 0 Ts 0;
                        0 1 0 Ts;
                        0 0 1 0;
                        0 0 0 1];

    % additive discrete noise covariance
    model.Q = @(x, Ts) q * [Ts^3 / 3,   0,          Ts^2 / 2,   0;
                            0,          Ts^3 / 3,   0,          Ts^2 / 2;
                            Ts^2 / 2,   0,          Ts,         0;
                            0,          Ts^2 / 2,   0,          Ts];

    % measurement function
    model.h = @(x) [1, 0, 0, 0;
                    0, 1, 0, 0] * x;

    % measurement function jacobian
    model.H = @(x) [1, 0, 0, 0;
                    0, 1, 0, 0];
    % additive measurement noise covariance
    model.R = r;
end
```

Task 5)

a)

```
% set parameters
q = 5 * eye(4);
r = 2 * eye(2);

% create the model and estimator object
model = discreteCVmodel(q, r);
ekf = EKF(model);

% initialize
%xbar(:, 1) = ...
%Pbar(:, : , 1) = ...

K_gain = [eye(2),           zeros(2, 2);
     (1/Ts) * eye(2), - (1/Ts) * eye(2)];
xhat(:, 1) = K_gain * [Z(:, 1); Z(:, 2)];
Phat(:, :, 1) = [r,         1/Ts * r;
                 1/Ts * r,  2/(Ts^2) * r + Ts/3 * eye(2)];

for k = 3:(K-1)
    % estimate
    [xp, Pp] = ekf.predict(xhat(:, k), Phat(:, :, k), Ts);
    xbar(:, k) = xp;
    Pbar(:, :, k) = Pp;
    % innovate
    [vk, Sk] = ekf.innovation(Z(:, k + 1), xp, Pp);
    % update
    [xupd, Pupd] = ekf.update(Z(:, k + 1), xp, Pp);
    xhat(:, k + 1) = xupd;
    Phat(:, :, k + 1) = Pupd;
end

% calculate a performance metric
RMSE = @(x, x_hat) (sqrt(mean((x' - x_hat').^2)));
posRMSE = RMSE(Xgt(1:2, :), xhat(1:2, :)); % position RMSE
velRMSE = RMSE(Xgt(3:4, :), xhat(3:4, :)); % velocity RMSE
```